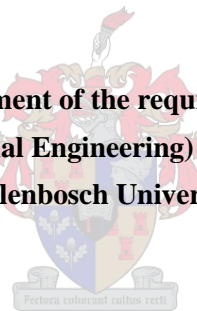


Implementation of machine learning techniques for railway wheel prognostics

by
Johannes Andreas Du Plessis

**Thesis presented in fulfilment of the requirements for the degree of
Master of Engineering (Industrial Engineering) in the Faculty of Engineering at
Stellenbosch University**



**Supervisor: Prof Cornelius J. Fourie
Co-supervisor: Prof A.F. van der Merwe**

April 2019

*The financial assistance of the PRASA Engineering Research Chair towards this research is hereby
acknowledged. Opinions expressed and conclusions arrived at, are those of the author and are not
necessarily to be attributed to PRASA*

Declaration

By submitting this thesis electronically, I declare that the entirety of the work contained therein is my own, original work, that I am the sole author thereof (save to the extent explicitly otherwise stated), that reproduction and publication thereof by Stellenbosch University will not infringe any third party rights and that I have not previously in its entirety or in part submitted it for obtaining any qualification.

Date: April 2019

Copyright © 2019 Stellenbosch University

All rights reserved

Abstract

The Passenger Rail Agency of South Africa (PRASA) is in the process of moving from a mostly reactive to a preventive approach to maintenance. The key to cost-efficient preventive maintenance strategies is the ability to predict the condition of components at a future time. The objective of this research project was to ascertain whether machine learning techniques can be used to provide prognostic predictions with respect to the condition of PRASA's railway and train components. The input data used to build the machine learning models was provided by Metrorail, a subsidiary of PRASA. Metrorail's railway wheels were selected to serve as the case study for this project, owing to the fact that the condition monitoring data collected on the railway wheels represented the most granular and complete data set related to fluctuating conditions of a Metrorail train component.

Five types of wheel wear are monitored by Metrorail. These forms of wheel wear are flange height increase, tread diameter decrease, hollow wear, flange slope increase and flange thickness decrease. Three machine learning models were built to provide prognostic predictions related to these types of wheel wear. These model types were logistic regression, artificial neural networks and random forest. One of each of these model types was developed for each of the wheel wear types. The performance of the models was then compared to ascertain which model performed the best for each of the wheel wear types. A normalised combination of sensitivity, specificity, F1 score and AUC was used to rank the models.

Logistic regression was surpassed by the artificial neural network and random forest models for each of the wheel wear types. The artificial neural network was the best prognostic model for tread diameter decrease (accuracy: 96.4%, normalised score: 0.964). Random forest was the best prognostic model for flange height increase (accuracy: 93.5%, normalised score: 0.822), hollow wear (accuracy: 92.5%, normalised score: 0.731), flange slope increase (accuracy: 94.2%, normalised score: 0.953) as well as flange thickness decrease (accuracy: 92.9%, normalised score: 0.733).

The encouraging results of these models showed that machine learning techniques can indeed be used to provide PRASA with train component wear prognostics. The models developed during the completion of this project can also be implemented by Metrorail to alleviate the need for manual wheel condition monitoring, by providing technicians with wheel prognostics.

Opsomming

Die Passasiers Spooragentskap van Suid-Afrika (PRASA) is besig om te beweeg vanaf 'n hoofsaaklik reaktiewe na 'n voorkomende benadering tot die onderhoud van bates. Dit is noodsaaklik om in staat te wees om die toekomstige toestand van bates te kan voorspel, sodat 'n koste-effektiewe benadering tot die onderhoud daarvan geïmplementeer kan word. Die doel van hierdie navorsingsprojek was om vas te stel of masjienleertegniese gebruik kan word om prognostiese voorspellings te maak ten opsigte van die toekomstige toestand van PRASA se treinonderdele. Die insetdata vir die masjienleermodelle was verskaf deur Metrorail, 'n filiaal van PRASA. Metrorail se treinwiele was gebruik as die gevallestudie vir hierdie navorsingsprojek, aangesien dit die treinonderdeel is met die mees volledige en gedetailleerde datastel, waarin die toestand van die onderdeel oor 'n bepaalde tydperk opgeneem is.

Drie masjienleermodelle was gebou om prognostiese voorspellings te gee ten opsigte van vyf vorms van wielverwering wat gemonitor word deur Metrorail. Die vorms van wielverwering is flenshoogte toename, wioldiameter afname, holverwering, flenshelling toename en flensdikte afname. Die drie masjienleermodelle was logistieke regressie, kunsmatige neurale netwerke en "random forest". Een van elk van hierdie modelle was gebou vir elkeen van die wielverweringstipes. Die voorspellingsvermoë van die modelle was dan met mekaar vergelyk om te bepaal watter model die beste geskik is om prognostiese voorspellings te maak vir watter wielverweringstipe. 'n Genormaliseerde kombinasie van akkuraatheid, sensitiviteit, spesifisiteit, F1 telling asook area onder kurwe was gebruik om te bepaal watter model die beste geskik was om prognostiese voorspellings te maak vir 'n gegewe wielverweringstipe.

Logistieke regressie as voorspellingsmodel het die swakste gevaar ten opsigte van elk van die wielverweringstipes. Kunsmatige neurale netwerke was die beste geskik vir wioldiameter afname prognose (akkuraatheid: 96.4%, genormaliseerde telling: 0.964). Die "random forest" was die modeltipe wat die beste presteer het ten opsigte van flenshoogte toename (akkuraatheid: 93.5%, genormaliseerde telling: 0.822), holverwering (akkuraatheid: 92.5%, genormaliseerde telling: 0.731), flenshelling toename (akkuraatheid: 94.2%, genormaliseerde telling: 0.953) asook flensdikte afname (akkuraatheid: 92.9%, genormaliseerde telling: 0.733).

Die hoogs positiewe resultate wat die modelle gelewer het, toon dat masjienleer beslis gebruik kan word om prognostiese voorspellings te maak met betrekking tot die toestand van PRASA se treinonderdele. Die modelle wat gebou was deur die verloop van hierdie navorsingsprojek kan ook geïmplementeer word deur Metrorail om prognostiese wielverweringsvoorspellings aan Metrorail se onderhoudstaakspanne te verskaf.

Table of Contents

| | |
|--|------|
| Declaration | ii |
| Abstract | iii |
| Opsomming | iv |
| List of Figures | viii |
| List of Tables | x |
| 1 Introduction | 1 |
| 1.1 Research problem | 2 |
| 1.2 Research objectives | 3 |
| 1.3 Scope and limitations | 4 |
| 1.4 Research methodology | 5 |
| 1.5 Chapter layout | 6 |
| 1.6 Chapter summary | 6 |
| 2 Case study introduction | 7 |
| 2.1 Background information of Metrorail's railway wheels | 7 |
| 2.1.1 Importance to Metrorail | 7 |
| 2.1.2 Railway wheel profile and degradation measurement | 8 |
| 2.1.2.1 Railway wheel component definitions | 9 |
| 2.1.2.2 Common railway wheel wear patterns | 10 |
| 2.1.2.3 Wheel wear measurement | 10 |
| 2.1.3 Motivation for selection as case study | 14 |
| 2.2 Chapter summary | 15 |
| 3 Literature review | 16 |
| 3.1 Overview of maintenance approaches | 16 |
| 3.1.1 Run-to-Failure maintenance | 17 |
| 3.1.2 Time-dependent maintenance | 18 |
| 3.1.3 Condition-based maintenance | 20 |
| 3.1.3.1 Experience-based prognostics | 23 |
| 3.1.3.2 Physics-based prognostics | 24 |
| 3.1.3.3 Data-driven prognostics | 25 |
| 3.2 Machine Learning | 26 |
| 3.2.1 Types of ML | 27 |
| 3.2.1.1 Supervised learning | 27 |
| 3.2.1.2 Reinforcement learning | 28 |
| 3.2.1.3 Unsupervised learning | 28 |
| 3.2.2 Classification versus regression | 29 |
| 3.2.3 ML as a search | 29 |
| 3.2.4 ML models | 31 |
| 3.2.4.1 Logistic Regression | 31 |

| | |
|--|----|
| 3.2.4.2 Artificial Neural Networks | 34 |
| 3.2.4.3 Random Forest | 40 |
| 3.2.5 Hyperparameter tuning and model evaluation | 43 |
| 3.2.5.1 Grid-search | 43 |
| 3.2.5.2 ML Model Performance Evaluation | 44 |
| 3.2.5.3 K-Fold cross-validation | 48 |
| 3.3 Chapter summary | 53 |
| 4 ML model development | 54 |
| 4.1 Data preparation | 55 |
| 4.1.1 Data read in and high level summary | 55 |
| 4.1.2 Data formatting and imputation | 56 |
| 4.1.3 Outlier correction | 57 |
| 4.1.3.1 First phase outlier removal | 57 |
| 4.1.3.2 Wheel instance identification | 57 |
| 4.1.4 Feature engineering | 67 |
| 4.1.4.1 Time between measurements | 68 |
| 4.1.4.2 Wheel instance age | 68 |
| 4.1.4.3 Relative FH difference | 68 |
| 4.1.4.4 Moving average wear rate | 69 |
| 4.1.4.5 Month of year | 69 |
| 4.1.4.6 Previous measure value | 69 |
| 4.1.5 Feature normalisation | 70 |
| 4.1.6 Target variable creation | 71 |
| 4.2 ML model building | 72 |
| 4.2.1 Train and test data set splitting | 72 |
| 4.2.2 Logistic regression model | 73 |
| 4.2.2.1 FH model evaluation | 73 |
| 4.2.2.2 TD Model evaluation | 74 |
| 4.2.2.3 HW model evaluation | 75 |
| 4.2.2.4 FS model evaluation | 76 |
| 4.2.2.5 FT model evaluation | 77 |
| 4.2.2.6 Logistic regression summary | 78 |
| 4.2.3 ANN model | 79 |
| 4.2.3.1 Hyperparameter grid creation | 79 |
| 4.2.3.2 Hyperparameter tuning | 80 |
| 4.2.3.3 Final model training and evaluation | 82 |
| 4.2.3.4 ANN performance summary | 87 |
| 4.2.4 Random forest model | 88 |
| 4.2.4.1 Hyperparameter grid creation | 88 |
| 4.2.4.2 Hyperparameter tuning | 89 |
| 4.2.4.3 Final model training and evaluation | 90 |
| 4.2.4.4 Random forest performance summary | 95 |

| | |
|--|-----|
| 4.3 Chapter summary | 96 |
| 5 Final model selection | 97 |
| 5.1 Model score combination | 97 |
| 5.2 Final model scores and selection | 98 |
| 5.3 Chapter summary | 100 |
| 6 Conclusion | 101 |
| 7 Recommendations | 103 |
| References | 105 |
| Appendix A | 108 |
| Data Preparation and Modelling Code Sample | 108 |
| Appendix B | 147 |
| Variable conversion from long to wide Format | 147 |

List of Figures

| | |
|---|----|
| Figure 1: Photo of the cross section of a tired railway wheel, adapted from [13] | 8 |
| Figure 2: Illustration of railway wheel components, adapted from [14] and [15] | 9 |
| Figure 3: Illustration of wheel and rail contact interface [16] | 10 |
| Figure 4: <i>MiniProf</i> mounted on a railway wheel | 11 |
| Figure 5: Railway wheel wear measurement schema | 12 |
| Figure 6: Illustration of numbering convention of motor coach wheels | 13 |
| Figure 7: Physical asset reliability bathtub curve [28] | 18 |
| Figure 8: Illustration of prognostic approaches, adapted from [34] | 23 |
| Figure 9: Conceptual framework of Supervised ML problem, adapted from [18] | 30 |
| Figure 10: Generic shape of the logistic function (Eq. 1.2) | 32 |
| Figure 11: Illustration of two connected neurons, Adapted from [40] | 34 |
| Figure 12: McCulloch-Pitts neuron model, adapted from [37] | 35 |
| Figure 13: Feedforward ANN | 36 |
| Figure 14: Visual representation of decision tree, adapted from [38] | 41 |
| Figure 15: Visual representation of Grid-search surface with respect to model performance, adapted from [45] | 44 |
| Figure 16: Example of an ROC curve, adapted from [47] | 47 |
| Figure 17: Illustration of model performance measured on training set (blue) vs. test set (orange) with ideal training stoppage point indicated (red) | 49 |
| Figure 18: Illustration of hold-out set model validation process | 51 |
| Figure 19: Illustration of K-fold cross-validation process | 52 |
| Figure 20: ML model development process diagram | 54 |
| Figure 21: Raw FH sample measurements with anomalies indicated | 58 |
| Figure 22: Cleaned FH sample measurements without wheel instance identification | 59 |
| Figure 23: Cleaned FH sample measurements after wheel instance identification | 60 |
| Figure 24: Box plot of average weekly FH wear of data sample | 61 |
| Figure 25: Box plot of negative consecutive FH measurement differences of data sample | 63 |
| Figure 26: Original FH vs. fixed FH for data sample 1 | 64 |
| Figure 27: Fixed FH for data sample 1 grouped by wheel instance | 65 |
| Figure 28: Original FH vs. fixed FH for data sample 2 | 65 |
| Figure 29: Fixed FH for data sample 2 grouped by wheel instance | 66 |
| Figure 30: Original FH vs. fixed FH for data sample 3 | 66 |
| Figure 31: Fixed FH for data sample 3 grouped by wheel instance | 67 |
| Figure 32: Example of ML model error contour plots for non-normalised and normalised input features [50] | 70 |
| Figure 33: ROC curve for logistic regression model of FH wear prognostics | 74 |
| Figure 34: ROC curve for logistic regression model of TD wear prognostics | 75 |
| Figure 35: ROC curve for logistic regression model of HW wear prognostics | 76 |
| Figure 36: ROC curve for logistic regression model of FS wear prognostics | 77 |
| Figure 37: ROC curve for logistic regression model of FT wear prognostics | 78 |
| Figure 38: FH ANN hyperparameter grid search output | 81 |
| Figure 39: ROC curve for ANN model of FH wear prognostics | 83 |
| Figure 40: ROC curve for ANN model of TD wear prognostics | 84 |

| | |
|---|-----|
| Figure 41: ROC curve for ANN model of HW wear prognostics..... | 85 |
| Figure 42: ROC curve for ANN model of FS wear prognostics..... | 86 |
| Figure 43: ROC curve for ANN model of FT wear prognostics | 87 |
| Figure 44: ROC curve for random forest model of FH wear prognostics | 91 |
| Figure 45: ROC curve for random forest model of TD wear prognostics | 92 |
| Figure 46: ROC curve for random forest model of HW wear prognostics | 93 |
| Figure 47: ROC curve for random forest model of FS wear prognostics..... | 94 |
| Figure 48: ROC curve for random forest model of FT wear prognostics..... | 95 |
| Figure 49: Suggested framework for ML model implementation..... | 103 |

List of Tables

| | |
|---|----|
| Table 1: Railway wheel wear measurement schema | 12 |
| Table 2: Metrorail wheel condition monitoring data schema | 13 |
| Table 3: List of discussed ML model hyperparameters | 43 |
| Table 4: Confusion Matrix Layout | 45 |
| Table 5: Initial summary of wheel data | 55 |
| Table 6: Confusion matrix for logistic regression model of FH wear prognostics | 73 |
| Table 7 :Confusion matrix metrics for logistic regression model of FH wear prognostics | 74 |
| Table 8: Confusion matrix for logistic regression model of TD wear prognostics | 74 |
| Table 9: Confusion matrix metrics for logistic regression model of TD wear prognostics | 75 |
| Table 10: Confusion matrix for logistic regression model of HW wear prognostics | 76 |
| Table 11: Confusion matrix metrics for logistic regression model of HW wear prognostics | 76 |
| Table 12: Confusion matrix for logistic regression model of FS wear prognostics | 77 |
| Table 13: Confusion matrix metrics for logistic regression model of FS wear prognostics | 77 |
| Table 14: Confusion matrix for logistic regression model of FT wear prognostics | 78 |
| Table 15: Confusion matrix metrics for logistic regression model of FT wear prognostics | 78 |
| Table 16: ANN hyperparameter grid | 80 |
| Table 17: Best performing ANN hyperparameters | 82 |
| Table 18: Confusion matrix for ANN model of FH wear prognostics | 82 |
| Table 19: Confusion matrix metrics for ANN model of FH wear prognostics | 83 |
| Table 20: Confusion matrix for ANN model of TD wear prognostics | 83 |
| Table 21: Confusion matrix metrics for ANN model of TD wear prognostics | 84 |
| Table 22: Confusion matrix for ANN model of HW wear prognostics | 84 |
| Table 23: Confusion matrix metrics for ANN model of HW wear prognostics | 85 |
| Table 24: Confusion matrix for ANN model of FS wear prognostics | 85 |
| Table 25: Confusion matrix metrics for ANN model of FS wear prognostics | 86 |
| Table 26: Confusion matrix for ANN model of FT wear prognostics | 86 |
| Table 27: Confusion matrix metrics for ANN model of FT wear prognostics | 87 |
| Table 28: Random forest hyperparameter grid | 89 |
| Table 29: Best performing random forest hyperparameter grid | 90 |
| Table 30: Confusion matrix for random forest model of FH wear prognostics | 90 |
| Table 31: Confusion matrix metrics for random forest model of FH wear prognostics | 91 |
| Table 32: Confusion matrix for random forest model of TD wear prognostics | 91 |
| Table 33: Confusion matrix metrics for random forest model of TD wear prognostics | 92 |
| Table 34: Confusion matrix for random forest model of HW wear prognostics | 92 |
| Table 35: Confusion matrix metrics for random forest model of HW wear prognostics | 93 |
| Table 36: Confusion matrix for random forest model of FS wear prognostics | 93 |
| Table 37: Confusion matrix metrics for random forest model of FS wear prognostics | 94 |
| Table 38: Confusion matrix for random forest model of FT wear prognostics | 94 |
| Table 39: Confusion matrix metrics for random forest model of FT wear prognostics | 95 |
| Table 40: Wheel wear measurement type target variable class counts | 98 |
| Table 41: Combined model scores for FH | 99 |
| Table 42: Combined model scores for TD | 99 |
| Table 43: Combined model scores for HW | 99 |

| | |
|--|-----|
| Table 44: Combined model scores for FS | 99 |
| Table 45: Combined model scores for FT | 99 |
| Table 46: Combined model scores for FT | 100 |
| Figure 49: Suggested framework for ML model implementation | 103 |
| Table 47: Example of long format variable | 147 |
| Table 48: Example of wide format variable | 147 |

1 Introduction

Members of a populace who are deprived of a means of mobility will undoubtedly miss out on social as well as economic opportunities, and will find it more difficult to reach employment, educational and healthcare facilities. Public transport plays a key role in providing communities with the mobility they require. However, the benefits of public transport include more than just the provision of mobility, and its beneficiaries include not only its consumers, but the community at large. A report by the Federal Highway Administration of the United States of America stated that public transport expands business opportunities which, in turn benefit public transport consumers, who gain access to new products and services as well as reduced prices due to increased market competition. It further benefits the businesses, which gain access to a new market. The report further stated that public transport helps to reduce road congestion, air pollution and energy consumption, all of which clearly benefit both consumers and non-consumers of public transport. Finally, the report stated that public transport is considered to be a crucial boon for the less fortunate community, by supplying a form of mobility in emergency situations that call for immediate evacuation [1]. A study of the benefits of public transport in the United States of America, specifically in relation to rail transit services, came to the same conclusions. This study further found that the availability of rail transit services correlated with lower per capita transit expenditure, lower per capita traffic fatalities, and a reduction in the average portion of household income devoted to transport [2].

The degree to which public transport benefits society is dependent on the number of people who find the service appealing enough to choose it over private transport. Lauren et al. investigated the attributes of public transport that drive the decision to use public transport instead of private automobiles. The investigation found that punctuality, service frequency, comfort and pricing numbered among the most prominent attributes that were considered by the public when comparing public transport to private automobile usage [3]. All of these attributes, as well as a myriad of other aspects, are affected by how well the equipment and physical assets involved in the provision of public transport are maintained. For instance, unplanned equipment downtime due to poor maintenance practices is detrimental to both service punctuality and frequency. Bad maintenance, both in the form of over-and undermaintenance, incurs unnecessary costs that has implications on the price of the service. Finally, it is easily conceivable that when asset wear is not restrained, it could result in deteriorating passenger comfort. Effective maintenance is therefore crucial to the success of public transport. What constitutes effective maintenance,

however, is not self-evident and is a concept that is continuously changing in response to advances in technology and maintenance techniques.

Machine learning (ML) is a relatively new technology that stems from a variety of academic disciplines, most notably computer science and statistics, that provides the means to automatically learn patterns from data and it has become practically ubiquitous as data storage and computing power has become cheaper [4]. It has found uses in a multitude of diverse fields ranging from marketing to astronomy, and physical asset maintenance is no exception [5], [6]. It stands to reason that public transport providers, as well as the public in general, can benefit from machine learning if it can improve the efficiency of public transport maintenance efforts [2], [7].

In South Africa, Metrorail, which is a subsidiary of the Passenger Rail Agency of South Africa, is responsible for the provision of commuter rail services in the country's main metropolitan areas. Metrorail serves over two million commuters daily, which represents nearly 15% of the national public transport market share [8]. Service excellence forms part of Metrorail's mission, and it is defined as follows:

"[Provision of a commuter rail service with] superior performance that is safe, reliable and affordable, and which makes a lasting impression by actively building brand loyalty – both internally (employees) and externally (customers) – ultimately adding benefit to the passenger" [9].

Metrorail's mission clearly aims to align the organisation with that which the public generally looks for in a transportation service. The reasoning for this is that all the aforementioned benefits of public transport will come to fruition if Metrorail can accomplish this mission. It therefore holds that effective maintenance underpins all that Metrorail aims to achieve. The question of whether machine learning can help improve Metrorail's maintenance efforts is one that certainly warrants investigation.

1.1 Research problem

Metrorail seeks to improve their maintenance efficiency in order to accomplish their mission, which is to provide a safe, reliable and affordable rail transportation service to South Africans. A key requirement for efficient maintenance is to carry out maintenance exactly when it is needed, and to the extent that it is needed. Unfortunately, Metrorail makes use of a fixed maintenance intervention frequency, which makes this task difficult, due to the fact that this form of

maintenance does not necessarily suit all components involved in service delivery. This results in some components being overly monitored and maintained, while others are neglected.

From a maintenance efficiency point of view, time spent on unnecessarily monitoring the condition of a component is seen as wastage. This similarly applies to the costs incurred and resources spent due to unexpected system downtime, repairs and replacements. This suggests that there is a balance that needs to be struck between under- and overmaintenance. It is immensely difficult for Metrorail to formulate a maintenance policy which will insure that maintenance operations will always be conducted at this point of equilibrium. This is due to the fact that the condition of their assets is influenced by a slew of ever-changing and often misunderstood factors.

By providing the means to predict the future state of an asset's condition, machine learning can provide maintenance managers with the information and insights that are required to implement a sustainable 'just-in-time' maintenance approach. Further advantages include improved supply chain planning and more efficient resource allocation. However, in order to implement machine learning for asset condition prediction, it requires the availability of data that tracks asset degradation over time, along with variables that could possibly indicate the causes of degradation. The problem addressed by this thesis is whether it is possible for Metrorail to implement machine learning techniques to predict the future condition of their assets, with the idea that this will aid in improving their maintenance efficiency.

1.2 Research objectives

The main objective of this thesis is to serve as a proof of concept for the use of machine learning techniques to provide Metrorail's maintenance management with physical asset prognostics. This objective is divided into sub-objectives which, once completed, will culminate in the completion of the main objective. These sub-objectives are provided in the following list, along with the chapter within which the sub-objective is addressed:

- Chapter 2
 - Motivate the necessity for railway wheel prognostics
 - Motivate why the selected case is suitable for implementation of machine learning techniques for physical asset prognostics
- Chapter 3
 - Provide literature review on approaches to physical asset maintenance
 - Review the advantages of data-driven predictive maintenance
 - Provide a description of how machine learning models are developed

- Provide a description of how machine learning model performance is evaluated
- Chapter 4
 - Provide a description of the methods used to prepare the railway wheel data obtained from Metrorail
 - Provide a description of the process followed to develop the machine learning models
 - Report on the performance of the machine learning models
- Chapter 5
 - Provide a description of, and motivate the process by which the performance of the models was compared
 - Identify the best performing model in the context of the case study
- Chapter 6
 - Draw conclusions from the results attained by the models
 - Ascertain as to whether the research problem has been addressed
- Chapter 7
 - Provide recommendations based on the drawn conclusions
 - Provide suggestions for future research

1.3 Scope and limitations

The purpose of this sub-section is to define the scope and limitations of the research presented in this thesis. These aspects of the research are provided in the following list:

- Due to time limitations, the research focused on three machine learning model types, each being a member of a different family of models
- The scope of the data set used in the case study was limited by what was made available by Metrorail at the time
- The range of hyper-parameters evaluated during the model development process was limited due to time and computing power limitations
- The *R* programming language was used for data processing and model development during the course of this research
- The results presented in this thesis are only relevant to the focal case study of the research, and might vary for different cases
- Although the models developed during the course of this research can be implemented in a real world setting, the intention is only to provide a proof-of-concept for the use of these models in the manner proposed in this thesis

1.4 Research methodology

The methodology that is followed to achieve the objectives as defined in Section 1.2 is described here. This research includes a survey of the theoretical aspects of various topics, as well as the application of these theories in a real world case study. This research was conducted using a pragmatic, quantitative approach. It is deemed pragmatic, due to the fact that it not only includes a survey of various theories, but also focuses on their implementation in a real world case study. The research is also deemed quantitative, because the performance of the developed models and, therefore, the success of the researched approach to physical asset prognostics is measured in quantitative terms.

In terms of the case study of the research, the focal component will be Metrorail's motor coach wheels and their degradation, which is frequently monitored during maintenance interventions. This component was selected because of the long history of motor coach wheel degradation measurements, that have been captured by Metrorail over time. This research will be conducted according to the following steps:

- Description of the selected case study, namely, motor coach train wheel degradation
 - Description of wheel shape, function and modes of degradation
 - Description of wheel degradation data captured by Metrorail
- An in-depth literature review on:
 - Physical asset maintenance, various approaches to physical asset maintenance and their advantages and disadvantages
 - The definition and conceptualisation of machine learning and the main types of machine learning
 - The workings of three machine learning model types, namely logistic regression, artificial neural networks and random forest models
 - Machine learning model tuning
 - Machine learning model validation
- Wheel degradation data preparation
 - Data cleaning, formatting and enrichment
- Application of knowledge acquired through literature review to proof of concept case study, specifically:
 - Motivation for data-driven prognostics for maintenance efficiency improvement with respect to selected Metrorail asset
 - Development of three machine learning models, covered in the literature review, on the prepared wheel degradation data
 - Evaluation and comparison of models' performances

1.5 Chapter layout

This thesis is structured in such a way that the reader is guided through the methodology described in Section 1.4, which is followed in order to achieve the objectives as defined in section 1.2. The chapter layout is as follows:

Chapter 1 introduces and justifies the topic of the study. The research problem is framed, the research objectives are defined as well as the limitations and methodology of the research.

Chapter 2 consists of an overview of the case study, namely the condition monitoring and deterioration modes of Metrorail's motor coach wheels.

Chapter 3 consists of an in depth literature review which is focused on the topics of physical asset maintenance and machine learning model development.

Chapter 4 presents the development of the machine learning models which will be used to predict the condition of Metrorail's motor coach wheels

Chapter 5 presents the validation results of the developed machine learning models

Chapter 6 presents a conclusion of the study

Chapter 7 consists of the final recommendations of the study, and provides suggestions for future research.

1.6 Chapter summary

This chapter serves as the introduction to this thesis. It provides an introduction to the problem addressed in this study and presents the formal research problem. The research objectives of the thesis are presented, with the overarching objective being to provide a proof-of-concept for the use of machine learning techniques for railway wheel prognostics at Metrorail. This chapter also presents the scope and limitations of the research presented in this thesis, as well as the research design and methodology that was followed. Finally, the chapter layout of this thesis is presented. The following chapter provides an overview of the case study for this thesis.

2 Case study introduction

As stated in Chapter 1, the overarching goal of this project is to serve as a proof of concept for the use of ML to improve Metrorail's maintenance efficiency by providing maintenance teams with physical asset prognostics. Metrorail's motor coach wheels were selected to serve as the proof of concept case study. The purpose of this chapter is to provide background information of this component in terms of its significance (in the context of Metrorail), its design, modes of degradation, condition measurement, as well as the reasoning behind its selection as proof of concept case study for the purposes of this project.

2.1 Background information of Metrorail's railway wheels

2.1.1 Importance to Metrorail

The importance of motor coach wheels to Metrorail can be described in terms of Metrorail's mission, which is to provide a safe, affordable and reliable rail transportation service. Railway wheels, among others, are one of the most crucial components when it comes to ensuring the safety of rail transport. The wheels form the interface between the train and the rails. They are responsible for transmission of propulsive force to the rails and for guiding and keeping the train on the rails. Railway wheels play a key role in preventing train derailments, as demonstrated in an investigation conducted by Liu et al. on the causes of train derailments. The results showed that railway wheel defects was the fourth most-frequently occurring cause of derailments, accounting for 5.2% of the 4,352 observed derailments [10]. Defective railway wheels can result in increased frictional forces between the wheel and the rails, which not only exacerbates the deterioration of the wheel, but can cause damage to the rails which threatens the safety of all commuters that make use of the rail network [11].

Metrorail makes use of a tired railway wheel that consists of a wheel disk that is fitted with a steel sleeve, referred to as a tyre. A photo of the cross section of a tired railway wheel is shown in Figure 1, with the disk and tyre indicated. An advantage of this wheel type is that normal wheel wear only deteriorates the tyre, which can be replaced, whereas with solid wheels, the entire wheel would have to be replaced when they become worn out. However, tired wheel tyres are still expensive to replace, with an estimated replacement cost of R19800 per tyre [12]. Therefore, if these wheels are poorly maintained, and require frequent replacement, the incurred

cost will certainly have a negative impact on Metrorail and might lead to an increase in the cost of Metrorail's services.

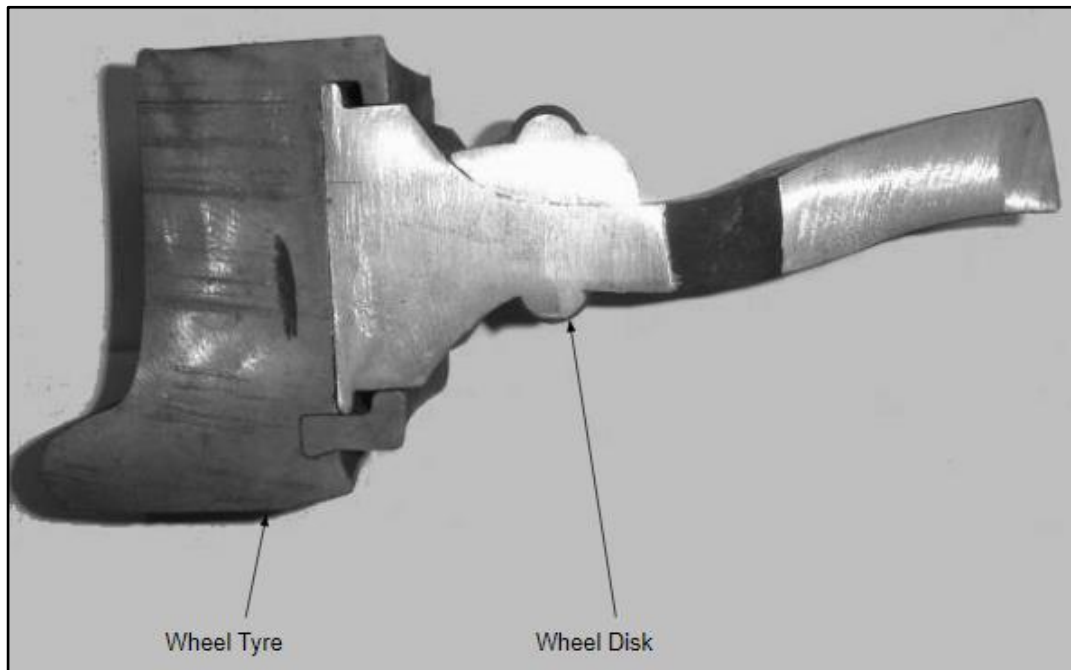


Figure 1: Photo of the cross section of a tyred railway wheel, adapted from [13]

Finally, in addition to Metrorail's reliance on railway wheels for service delivery, railway wheels pose a unique challenge to Metrorail from a reliability and service delivery perspective due to the fact that wheel repairs are outsourced. This means that Metrorail's maintenance management must coordinate with an external organisation and account for the external organisation's lead times and availability, in their wheel maintenance planning.

2.1.2 Railway wheel profile and degradation measurement

The focus of this subsection is to define the terminology that is used when referring to the various parts of Metrorail's motor coach wheels, and their modes of degradation. Firstly, the different components of these wheels will be defined. Secondly, the common wear patterns of motor coach wheel tyres will be described. Finally, the methods used by Metrorail to measure wheel wear will be described.

2.1.2.1 Railway wheel component definitions

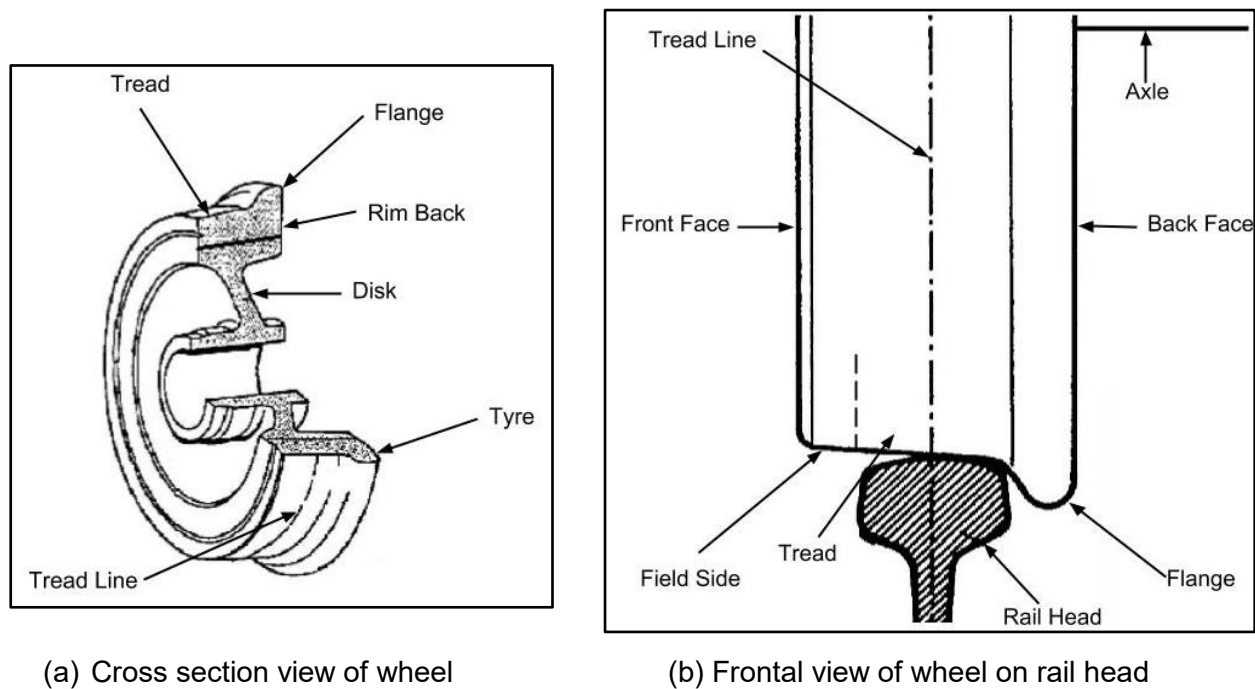


Figure 2: Illustration of railway wheel components, adapted from [14] and [15]

The running surface of railway wheels has a particular profile that allows the wheels to perform their intended functions, the most important of which is to keep the train on the railway line and to allow the train to turn. The tread and the flange are the two main sections of the running surface that are responsible for these two aforementioned wheel functions. An illustration of the cross section of a typical railway wheel as well as a railway wheel as it sits on a rail are provided in Figure 2 (a) and Figure 2 (b) respectively, with the some of the most notable sections indicated [14], [15].

The wheel sits with its tread on the rail and is slightly tapered from its back side towards the front side. This tapered wheel shape is what guides the train on the rails and allows a train to turn despite having solid wheel axles. The tread line is the line traced on the tread of a wheel by the expected contact point between the wheel and the rail when the train is moving in a straight line. This is the point on the wheel tread that is expected to deteriorate first, under normal operating conditions. The flange is located on the back face edge of the wheel's running surface and it runs on the inside edge of the rail head. The flange is a safety feature of the wheel and its role is to prevent the wheel from slipping off the rail head when the aforementioned self-steering effect of the wheel fails to guide the wheel on the rail.

2.1.2.2 Common railway wheel wear patterns

The common wear pattern on railway wheels corresponds to the contact interface between the wheel and the railway head. Figure 3 illustrates the running surface profile of an unworn wheel (blue) overlaid with that of a worn wheel (red) [16]. A rail head is also shown (green) to highlight the contact interface between the wheel and the rail.

As shown in Figure 3, the wear pattern gives rise to certain wheel features that can be measured to determine the extent of wheel deterioration. The main features are the following:

- Reduced flange thickness
- Increased relative height between flange tip and wheel tread
- Increased angle tangent to flange face
- Reduced wheel diameter due to wheel tread wear
- Hollowed areas on wheel tread

Metrorail measure these wheel features during wheel condition inspections to determine whether the wheel tyre should be decommissioned and repaired or replaced.

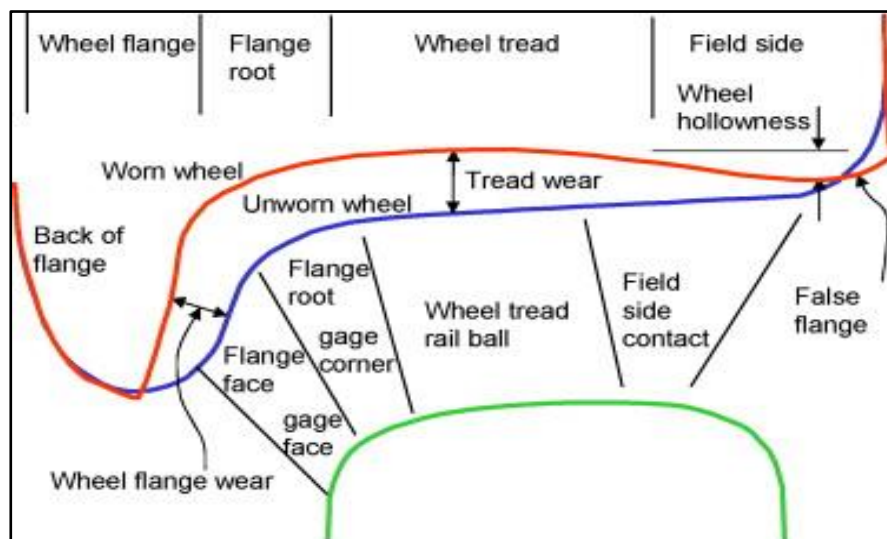


Figure 3: Illustration of wheel and rail contact interface [16]

2.1.2.3 Wheel wear measurement

Measuring the extent of the aforementioned types of wheel wear is challenging because of the smooth transition between the various components of a wheel's running surface. This makes it difficult to determine, for instance, where the flange root ends and the wheel tread begins. This is made even more challenging when wheel wear starts to deform the wheel as shown in Figure 3. Metrorail adopted a standardised procedure to measure the extent of wheel wear to address this issue. Some of the main advantages of adopting such a procedure are that it allows for

measurements to be repeated and compared, and it allows for the establishment of threshold values for each type of wear, that can be used to evaluate whether the wheel should be decommissioned. Such a standardised method also speeds up wheel condition monitoring.

To measure wheel profiles, Metrorail makes use of the *MiniProf* Wheel measurement system. The frontal and side view of a *MiniProf* mounted on a railway wheel are shown in Figure 4 (a) and Figure 4 (b), respectively. The *MiniProf* is a full contact measurement system that is magnetically mounted onto the flange of a railway wheel during condition monitoring. To take a profile measurement, a technician drags the magnetised contact wheel, which sits at the tip of the measuring probe, across the railway wheel's running surface. The *MiniProf* is programmed to relay predefined measurements to a database and notifies the technician when one of the measurements has crossed a wheel decommissioning threshold.



(a) Frontal view of *MiniProf* on wheel, [17]



(b) Side view of *MiniProf* on wheel

Figure 4: *MiniProf* mounted on a railway wheel

The *MiniProf* produces measurements that convey the extent of wear as per the five signs of wheel wear listed in Section 2.1.2.2. This is achieved by using a built-in reference wheel profile in conjunction with the measurement schema shown in Figure 5. The measurements are Flange Thickness (FT), Tread Diameter (TD), Flange Slope (FS), Flange Height (FH) and Hollow Wear (HW). FT is measured at 14 mm below the highest tip of the flange. TD is a calculation of the wheel's diameter at the point 82.5 mm from the back face. FS is measured as the horizontal distance between the point on the tread side of the flange that is 2 mm below the flange tip, and the point on the tread side of the flange that is 14 mm below the flange tip. A small FS

measurement indicates that the angle of the flange has increased. FH is the vertical distance between the flange tip and the point on the wheel tread where TD is measured. Finally, HW is measured as the maximum vertical height difference between the reference wheel tread and the measured wheel tread.

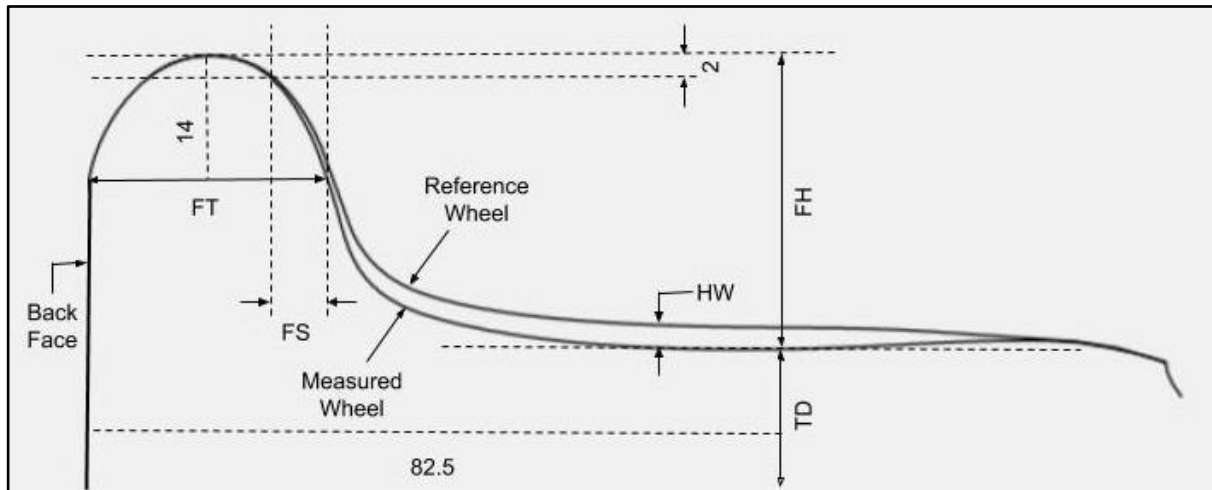


Figure 5: Railway wheel wear measurement schema

Table 1: Railway wheel wear measurement schema

| Measurement | Cut-off Value (mm) | Crossing Direction |
|-------------|--------------------|--------------------|
| FT | 19 | Decreasing |
| TD | 978 | Decreasing |
| FS | 6.5 | Decreasing |
| FH | 35 | Increasing |
| HW | -3 | Decreasing |

Each of the aforementioned measurements recorded by the *MiniProf* has a threshold value, which Metrorail has established, that is used to determine if a wheel should be decommissioned. Each of the wheel measurements is listed in Table 1 with their prospective cut-off values and the direction in which the measurement will cross the cut-off value due to wear.

At Metrorail, when a wheel measurement is taken with the *MiniProf* the measurements listed in Table 1 are recorded in a database table, along with additional details pertaining to both the wheel and the instance of the measurement itself. The schema of the database table as well as a short description of each field is shown in Table 2. The short descriptions are sufficient for all the table

columns save for Wheel Number, which is designated to a wheel according to a convention that is implemented by Metrorail. Each motor coach has a handbrake mechanism on one of its sides which is used as a reference point when numbers are assigned to the motor coach's wheels. As shown in Figure 6, wheel number 1 is found at the furthest end to the left when one is facing the side of the motor coach on which the handbrake mechanism is. From there on, wheel numbering follows a zig-zag pattern across the width of the motor coach as shown in Figure 6.

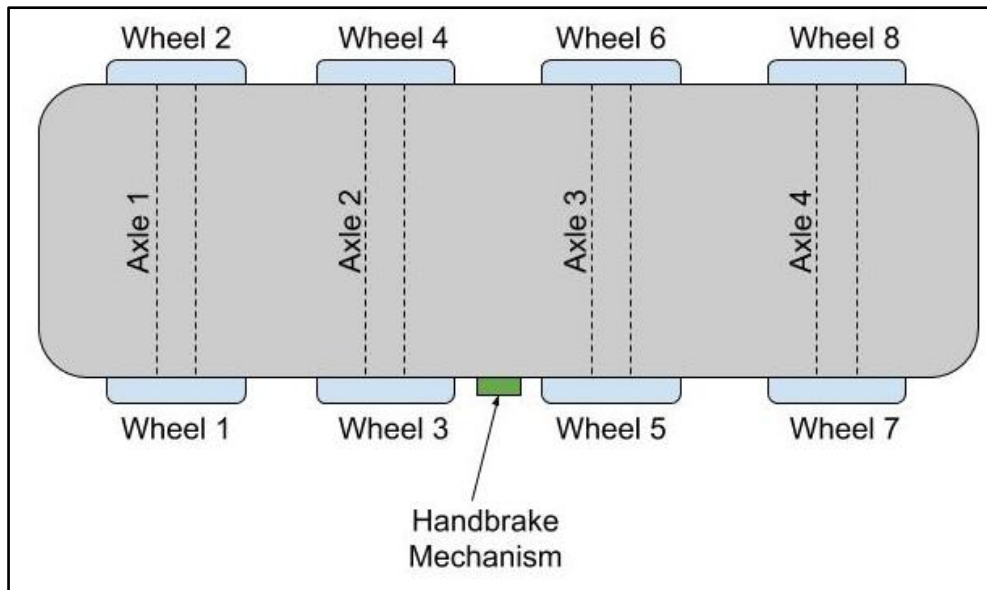


Figure 6: Illustration of numbering convention of motor coach wheels

Table 2: Metrorail wheel condition monitoring data schema

| Field Name | Field Description |
|--------------|---|
| ID | Unique identifier for the measurement |
| Date | Date the measurement took place |
| Time | Time the measurement took place |
| Stock | Motor coach model |
| Axle Number | Unique identifier for each axle on a bogie, as shown in Figure 6 |
| Bogie Number | Unique identifier for each bogie in a set |
| Wheel ID | Unique identifier for each wheel on a bogie, as shown in Figure 6 |
| FT | Measured flange thickness |
| TD | Measured tread diameter |
| FS | Measured flange slope |

| Field Name | Field Description |
|------------|------------------------|
| FH | Measured flange height |
| HW | Measured hollow wear |

2.1.3 Motivation for selection as case study

Ideally, a problem must satisfy three key requirements if one wishes to use supervised ML techniques to solve it. Firstly, there should be reason to believe that there is some process at work that gives rise to the problem. In other words, the problem should not occur purely randomly. Secondly, the problem should exhibit some randomness that prohibits one from formulating a mathematical function that describes the problem exactly. Finally, there should be enough data available that contain examples of positive and negative instances of the problem (in the binary classification case), as well as various explanatory variables from which the ML algorithm can derive an adequate model of the problem [18].

The first two requirements are not essential for the implementation of supervised ML techniques. For instance, one can use these techniques to model a purely random phenomenon, but the model will not perform any better than a random guess. One can also use supervised ML techniques to model a process that can be described exactly with a mathematical function, although the produced model will not perform any better than the mathematical function. However, the final requirement is essential for supervised ML, because the availability of training data sits at the core of supervised ML and without it there is nothing for the ML algorithms to learn from.

Metrorail's motor coach wheels were selected as a proof of concept case study for the use of supervised ML techniques to aid maintenance efforts, due to the fact that this component adheres to all three of the aforementioned requirements for using supervised ML to solve a problem. Firstly, there is reason to believe that motor coach wheel degradation is not a purely random process, rather, it is a result of various physical and chemical stressors that are related to various characteristics of the wheels and railway. As of yet, there is no mathematical formula that perfectly describes the rate of railway wheel degradation, because it is a stochastic process that involves some randomness that is introduced by characteristics of the individual wheels and the manner in which they were maintained and operated. Finally, Metrorail has a large data set comprised of records that contain the information listed in Table 2, that ML algorithms can use to develop models that predict future wheel conditions or rates of degradation. These characteristics of Metrorail's railway wheel degradation problem make it a case study that can serve as a proof of concept for the use of supervised ML techniques to aid Metrorail's maintenance efforts.

2.2 Chapter summary

This chapter provided the required background information of the selected case study that will serve as proof of concept for the use of ML to aid Metrorail's maintenance. Firstly, the importance of Metrorail's motor coach wheels was discussed and it was found that they play a crucial role when it comes to ensuring the safety of Metrorail's services. These wheels also impact on service reliability, availability as well as cost, ultimately affecting all aspects of Metrorail's mission. Secondly, the profile of the running surface of railway wheels was discussed. The components of the wheel as well as their function and wear patterns were described. Metrorail's approach to measuring wheel wear was also described in terms of the equipment used as well as the various conventions used when measuring wheel condition. Finally, the chapter was concluded with a motivation as to why Metrorail's motor coach wheels are a suitable candidate to serve as a proof of concept case study for the use of ML to aid Metrorail's maintenance efforts. The motivation was based on the requirements that a problem has to satisfy in order for it to be addressed with supervised ML techniques.

3 Literature review

3.1 Overview of maintenance approaches

Any enterprise that utilises physical assets to manufacture goods or deliver services will benefit greatly by maintaining those assets. Firstly, most physical assets used for service delivery or production of goods are subject to deterioration due to usage and age. Secondly, dependence on these physical assets will inherently call for them to be reliable. Maintenance can be defined as activities undertaken to conserve or restore the condition of a system or system component to within a defined operable specification [19], [20]. The goal of maintenance, in general, is to maximise the utilisable lifetime of equipment, while minimising the cost attributed to equipment reliability or lack thereof. The aforementioned concept, 'equipment lifetime', refers to the entire lifetime of equipment, i.e. time from implementation to decommissioning, as well as the mean time between equipment breakdown and the mean time of equipment downtime. If all three these aspects of equipment life are optimised then the utilisable lifetime of equipment will be optimised as well [21]. The relationship between maintenance and reliability is therefore clearly established if we take the definition of reliability into account. Here we define reliability as the probability that a system will satisfactorily perform its specified task for a specified length of time under specified environmental conditions [22].

The relationship between the goal of maintenance and reliability is explained as follows: If equipment is not sufficiently maintained, the mean time to failure is expected to decrease, whereas the expected downtime will increase. Therefore, the probability that the equipment will adequately perform its specified task for a given period of time, will inevitably decrease. The incurred cost attributed to the decreased equipment reliability is both that of the frequent repair or replacement of broken-down equipment as well as the production time that was lost during downtime. If maintenance is done too often, then equipment reliability will improve. However, the return on investment will diminish to the point where the cost incurred becomes unjustifiable [21]. These cases suggest that a set of rules should be put in place that dictate how maintenance should be conducted, in order to achieve a balance between reliability and cost. In most organisations these rules are formalised as a maintenance policy.

Some of the core decisions formulated in a maintenance policy are how regularly maintenance should be conducted, what signals the requirement for maintenance interventions, and the

extent to which equipment is maintained. A well-defined maintenance policy should satisfy both the technical requirements as well as the financial constraints of the organisation.

An exhaustive review of maintenance policies will be comprised of thousands of policies; however, most of them can be subsumed under a relatively small number of categories. These categories differ mainly in terms of what prompts a maintenance intervention and the extent to which an asset is restored during a maintenance intervention. In the following subsections the most prevalent categories of single-unit maintenance policies are reviewed [20].

Before reviewing the various asset maintenance policy classes, it is important to clarify what is meant when referring to physical assets. Assets can be classified as being either engineering assets or financial assets. In general an asset is defined as anything that can be owned by a legal entity, has value, and is widely regarded as legitimate currency for settlement of debt, payment of financial commitments, and inheritance [23]. The concept 'engineering asset' is defined in the same way, save for an additional requirement, which is that engineering assets have material existence, which is why they are also referred to as physical assets. Financial assets, on the other hand, only have to exist in a legal sense. Examples of physical assets are land and buildings, equipment and inventories. Financial assets generally take the form of legally binding contracts and agreements [24]. The terms 'physical assets', 'assets' and 'equipment' are used interchangeably in this literature review unless specified otherwise.

3.1.1 Run-to-Failure maintenance

With the *Run-to-Failure* (RTF) maintenance policy, no effort is made to anticipate or prevent physical asset failure. Rather, the asset is allowed to deteriorate until failure occurs, at which point the system is repaired or replaced [21], [25], [26]. This type of maintenance policy represents the simplest form of maintenance, and is suitable for systems for which the consequence of failure is negligible or the cost of system diagnostics is unjustifiably high.

The advantage of such a maintenance policy is that it incurs relatively low cost if implemented appropriately. Furthermore, this maintenance policy also calls for little analysis beyond the initial maintenance policy feasibility study. This is due to the fact that no continuous condition monitoring or failure analysis is required once it is decided that RTF maintenance will be applied for a physical asset. The only analysis that is required is to utilise time to failure data in order to improve maintenance management's estimates of the expected lifetime of an asset. It therefore requires little knowledge in terms of the physics of failure and failure prediction, and can be implemented by a small crew of technicians. Finally, RTF maintenance does not require frequent condition monitoring and maintenance interventions, which reduces the frequency with

which the assets are interrupted. The most notable disadvantages of corrective maintenance are that it only allows for maintenance planning in so far as knowledge is available regarding the expected lifetime of the physical asset. If highly accurate estimates of the expected lifetime of an asset is available, then this knowledge can be utilised by maintenance management for planning purposes. However, such knowledge is not always on hand and in these cases redundancies must be put in place to avoid operational downtime [27]. This form of maintenance becomes risky when applied to systems or equipment that are ill-suited for RTF maintenance. An example of such a system or piece of equipment would be one that provides an organisation with crucial functionality or that is immensely expensive or difficult to replace. If such a system is neglected until failure, the cost of downtime and replacement can easily outweigh the advantages of RTF maintenance [25].

3.1.2 Time-dependent maintenance

Under a *Time-dependent Maintenance* (TDM) policy, maintenance actions are taken, at predefined time intervals, that detect, preclude or reduce asset deterioration [25]. During unscheduled breakdowns a decision is made to either repair or replace the asset. TDM is the simplest maintenance policy class that introduces preventive maintenance by incorporating failure time analysis. The bathtub curve is often used to model the hazard of asset failure over time.

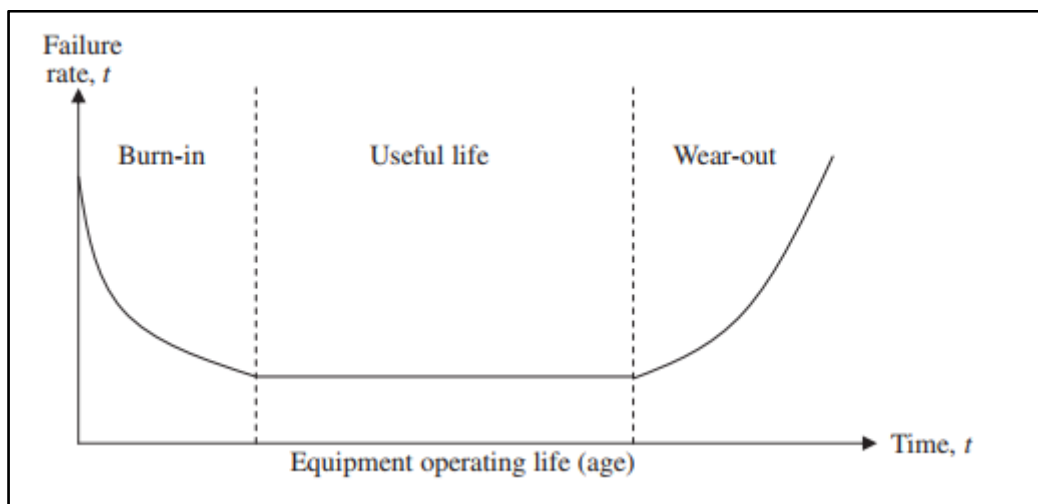


Figure 7: Physical asset reliability bathtub curve [28]

As seen in Figure 7 [28], the bathtub curve defines three stages of asset failure hazard over time. Initially the rate of failures is high due to 'infant mortality'. During this phase physical assets or components that contain manufacturing flaws resulting in early, sudden failures are weeded out. After an initial burn-in phase, failure rates decrease to a steady state. As the asset ages it

moves into the wear-out stage where failure rates start to increase due to deterioration. TDM aims to schedule maintenance interventions in such a way that useful life is extended by restoring assets prior to excessive wear-out.

TDM policies stemmed from a maintenance policy class known as *Periodic Replacement with Minimal Repair*, where an asset is replaced at fixed time intervals and minimal repair is conducted when failures occur at intervening times [20]. As the concern for balancing reliability with cost grew, the practice of imperfect maintenance became more established. Imperfect maintenance refers to maintenance that is not concerned with restoring assets to 'good-as-new' condition, but is rather focused on restoring assets to within tolerated specifications [29]. The introduction of imperfect maintenance allowed for the progression from asset replacement after a predefined time interval to preventive imperfect maintenance after a predefined time interval. Under the TDM policy the assumption is that replacement and repair incurs a larger cost than an imperfect maintenance intervention. This assumption is chiefly based on the fact that physical assets, such as machinery and equipment, often take the form of a system of interactive components. Failure of one of these components will most likely not be isolated to the component itself, rather, the failure will cause damage to various parts of the system, resulting in a sudden increase in the cost of fixing the damage after failure. Failures can also occur during times of high demand for the asset's function, incurring costs due to lost production time and product rejection. A maintenance job can focus efforts and expenditure to ensure that only the most vulnerable, deteriorated and crucial components are restored in order to avoid the cascading damage of catastrophic component failure.

A key competency required for implementing TDM, is the ability to select suitable time intervals between maintenance interventions. This process requires knowledge pertaining specifically to the assets of interest, the costs associated with repairing and replacing these assets as well as the ability to record and analyse these assets' failure time data. The question of how an asset's age is measured is also important when implementing TDM. For instance, one option is to use the number of hours since installation to measure age, another is to only count operational hours, while some might opt to measure an asset's age in the number of units it has produced. These skill and knowledge requirements limit the applicability of TDM for many enterprises. Furthermore, TDM also includes some built-in assumptions which should be validated prior to implementing this policy. Firstly, the TDM policy assumes that the assets of interest are repairable and that they will enter a wear-out phase where failure rates will increase with age. TDM also assumes that an asset's reliability is primarily a function of its age and that failure-time data is available to derive the relationship between age and reliability. All of the aforementioned assumptions must hold to implement TDM, save for the requirement for failure-time data.

However, if failure-time data is not available, an organisation will have to either rely on experience or manufacturer recommendations to establish appropriate time intervals between maintenance interventions. These aspects of TDM limit its applicability and are seen as disadvantages of TDM.

There are, however, many advantages to TDM. Some of the advantages of TDM are that maintenance can be scheduled to best suit production schedules. TDM also introduces the concept of preventive maintenance which improves reliability and reduces both unexpected downtime and the need for spare production equipment. With preventive maintenance, maintenance becomes a competitive activity that forms part of the production or service delivery process, as opposed to an accepted loss, as it is with RTF maintenance. TDM also provides many advantages compared to more advanced maintenance classes. The data required to implement TDM, namely failure-time data, is relatively easy to record. Furthermore, the analysis of TDM is less involved than the analysis of more complex data required by the more advanced maintenance classes [28].

3.1.3 Condition-based maintenance

Condition-based maintenance (CBM) is a maintenance policy class that was introduced in the 1970s in an attempt to improve the effectiveness of maintenance decisions. Under CBM, maintenance actions are recommended based on information obtained by analysing data collected during routine condition monitoring processes. Routine condition monitoring here refers to collecting data frequently, over an asset's lifetime, that describes the asset's operating conditions. The data is often multivariate, with typical variables including vibration measurements, operating temperature, measurements of contaminants in machine lubricant and noise emission [28], [30].

The rationale behind CBM is the fact that most asset failures are preceded by signals that indicate that a breakdown is imminent. Therefore, by frequently monitoring and analysing operating condition data, these signals can be detected and correct actions can be taken to avoid catastrophic failures. By implementing CBM, asset health is better defined, monitored and maintained. CBM also reduces the cost of maintenance as well as the rate of unexpected failures [28], [30].

Condition monitoring is the core decision driver of CBM. During this process, sensors are used to measure signals that communicate the condition of an asset. The goal of condition monitoring is twofold: Firstly, condition monitoring generates the data that drives maintenance decisions and secondly, it provides maintenance teams with data from which new insights into cause and

effect relationships between operating conditions and failure modes can be derived [28]. CBM can be classified based on whether condition monitoring is done offline or online, whether condition monitoring data is collected continuously or periodically, and according to the type of maintenance decision-making that is supported by the condition monitoring data.

Online condition monitoring is when condition monitoring occurs while the physical asset is operational while offline condition monitoring is condition monitoring that is done while the physical asset is idle or not running. Many incipient asset faults can only be detected while the asset is operational. It is therefore, often advantageous to implement online condition monitoring. On the other hand, many signs of asset degradation can only be observed when the asset is not operational. Therefore, the best approach is one that implements both offline and online condition monitoring for their respective fault types [28].

Periodic condition monitoring refers to condition monitoring that is done at fixed time intervals, for instance every hour, daily, or at the end of every shift. It is usually conducted manually, using hand-held sensors and instruments (such as decibel meters and Vernier calipers) and the operator's senses (an example of this would be visual evaluation of component cleanliness for instance), to assess the condition of the asset. Continuous condition monitoring refers to a condition monitoring system that continuously reports the current condition of the asset to an operator, or saves the data in a database for later evaluation. This form of condition monitoring is usually automated, relying on specialised equipment to measure and relay data to operators or centralised databases. A key disadvantage of continuous condition monitoring is that it is generally more expensive to implement than periodic condition monitoring due to the requirement for specialised equipment and process automation. The advantage of continuous condition monitoring is that it makes real-time failure prevention possible and it provides a more detailed history of the condition of the physical asset. This advantage is due to the relatively high frequency at which a typical continuous condition monitoring system measures the condition of a physical asset. This not only significantly reduces the chances of missing the signals that indicate imminent failure, but also provides data analysts with an abundance of detailed data to use for failure behaviour analysis and modelling [28], [30].

The range of informed maintenance decisions that is available to maintenance managers operating under CBM is directly dependent on the type of maintenance decisions that are supported by the data and insights generated by the condition monitoring process. In general, these maintenance decision support types can be divided into two classes, namely diagnostic and prognostic [28]. The Oxford Dictionary defines diagnosis as "the identification of the nature of an illness or other problem by examination of symptoms" [31]. The goal of a diagnostic

maintenance decision support system is to notify maintenance managers or operators when an asset is functioning abnormally. Furthermore, such a system must determine possible causes for the abnormality based on symptoms that are detectable in the condition monitoring data. By providing maintenance managers with an early warning of a possible failure as well as the nature of the approaching failure and the possible causes thereof, a diagnostic maintenance decision support system can aid in avoiding asset failures. Accurate and timely diagnostics can also dramatically decrease maintenance reaction times as well as downtimes due to maintenance. The disadvantage of a diagnostic decision support system stems from the fact that many assets can accomplish their tasks for a long period of time, despite abnormal operating conditions. Although a diagnostic system indicates that abnormal operating conditions are prevalent and warns machine operators and maintenance managers of the increased risk that a failure might occur, it provides no indication as to how long the asset will survive given its current condition. Given that an efficient maintenance policy is one where high levels of reliability are achieved by conducting maintenance interventions exactly when they are needed and exactly to the extent that they are needed, it is clear that a diagnostic decision support system by itself will not be sufficient for implementing such a maintenance policy. Although a diagnostic decision support system provides the latter of the two requirements, it lacks the former due to the fact that diagnostics provides no information regarding remaining useful asset life [28], [30].

One way to address the aforementioned lacking capability of a diagnostic decision support system is to implement a prognostic decision support system. A prognosis is defined as a forecast of the likely outcome of a situation [32]. In the context of physical asset maintenance, a prognostic decision support system allows maintenance managers to predict the future state of the condition of an asset based on its current and historic condition states and usage. As with a diagnostic decision support system, a prognostic decision support system aids in detecting abnormal operating conditions and identifying causes of incipient faults; however, unlike a diagnostic decision support system, a prognostic decision support system provides further information by predicting the Remaining Useful Life (RUL) of an asset [28], [33]. A prognostic decision support system should report a RUL estimate for each of the failure modes concerning the asset under consideration. Moreover, such a system should provide a measurement of the uncertainty associated with each of the RUL estimates. To do so, the prognostic decision support system must take into consideration both the condition monitoring data collected during normal as well as abnormal operating conditions, and the condition monitoring data that is concurrently recorded. Furthermore, the work scheduled for the asset within the time horizon of the prognostic assessment and the impact it has on the condition of the asset must be taken into account [34].

A prognostic maintenance decision support system provides a multitude of advantages to maintenance managers if implemented correctly, i.e. in such a way that the RUL estimations the system makes are accurate. Hatem et al. stated that a prognostic maintenance decision support system minimises asset downtime and therefore also increases productivity. It also reduces the necessity for redundancies and keeping spare parts on hand seeing as maintenance managers can plan when to order spare parts as needed based on RUL estimates. In general, it provides the means by which an organisation can move from a 'fail and fix' maintenance strategy to a predict and prevent maintenance strategy [34].

3.1.3.1 Experience-based prognostics

Prognostic maintenance decision support systems can be classified according to the means by which the system derives its prognoses and determines the RUL of the asset being monitored. In general, such a system will fall under, or borrow characteristics from one of the three prognostic approaches illustrated in Figure 8 [33].

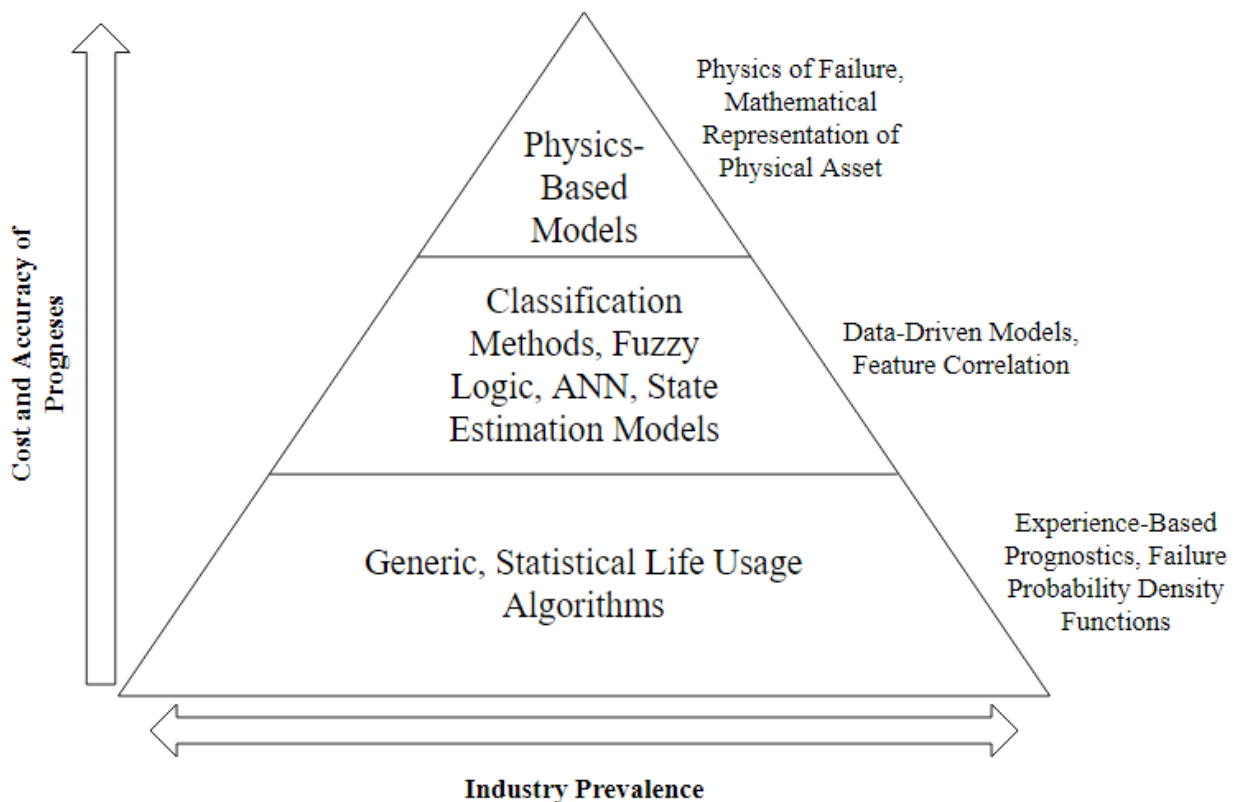


Figure 8: Illustration of prognostic approaches, adapted from [34]

The first of these can be referred to as the Experience-based approach. Experience-based prognostics is the simplest prognostic approach that is implemented on assets on which no

operating condition data is generated or recorded. The only information related to the asset that is used is the time that the asset has been operational along with descriptive statistics of the failure rates of a large population made up of the asset of interest. The failure rate information is often provided by the manufacturer of the asset or calculated using 'time to failure' measurements taken from previously implemented assets. The time to failure information is then used to calculate the mean time to failure for an asset. Preventive maintenance is planned and executed based solely on this estimated mean time to failure. In essence Experience-based prognostics is synonymous to TDM as described in Section 3.1.1.2 and therefore provides the same advantages and disadvantages as TDM.

3.1.3.2 Physics-based prognostics

Physical Models, or rather Physics-based prognostics, is situated at the top of the pyramid depicted in Figure 8, indicating that, of the three prognostic approaches, it is generally the most costly option and provides the most accurate prognoses. Physics-based prognostics entails the development of a mathematical representation of an asset, specifically with regards to the asset's various degradation processes and failure modes. Such a mathematical representation of a real-world entity is referred to as a model of the entity. With Physics-based prognostics, the model is developed from the physics theories and expressions that govern the asset's degradation and failure processes. To develop such a model of an asset requires, firstly, in-depth knowledge of the workings of the asset, its subcomponents, the environment in which it operates as well as the stressors to which it is exposed. Secondly, it requires the ability to express these elements and their impact on the condition of the asset in physics terms [34].

Once the model is developed, the operating conditions of the asset are monitored in such a way that the recorded data can be compared to readings that are produced by the model. Thus the model acts as a baseline for the operating conditions of the asset and by comparing the condition monitoring data with the model's outputs one can determine the current health of the asset. After establishing the current health of the asset, the model can be used to simulate what effects future workloads and environmental conditions will have on the condition and RUL of the asset, given the asset's current condition [34].

A crucial step in the process of developing a Physics-based model of an asset is to clearly define the scope of the model. The scope of the model should determine what asset components will be modelled and to what degree of detail they will be modelled. The model scope also determines what stressors, failure modes and degradation processes to model. Finally, the model scope should clearly demarcate which aspects of the operational environment of the asset to account for. If the model's scope is set too wide the model might represent the

actual asset extremely accurately, but its development will become unfeasibly complicated. Conversely, defining the model's scope too narrowly will result in a model that is trivial to develop but provides a very poor representation of the actual asset.

The advantages of Physics-based models are that they are very descriptive seeing as the models are based on the physics of the degradation and failure processes. In other words, Physics-based models can provide clear feedback, in physics terms, of what failure mode is most likely to occur next, how long the asset will survive given its current condition and operating environment, and which factors are aggravating the degradation of the asset. Physics-based models can also be made to be very accurate depending on the amount of detail that is permitted by the scope of the model. On the other hand, Physics-based models require a considerable amount of knowledge and skill to build. Physics-based models, and therefore Physics-based prognostics, are also generally more time-consuming and expensive to implement than Experience-based or Data-driven prognostics.

3.1.3.3 Data-driven prognostics

Often feasibility studies will indicate that the cost associated with developing a Physics-based model of an asset is unjustified. The reason for this can be due to the complexity of the physical asset under consideration that makes model development very difficult and costly, or that the physics of the degradation process are ill-understood and would therefore result in an inaccurate model that performs poorly, or that the asset is not valuable enough to justify the cost of a Physics-based model, to name a few. In such cases Data-driven modelling can be used as an alternative to Physics-based modelling [34], [4].

Data-driven modelling is a technique that allows asset maintenance teams to develop models that can infer the state of an asset's health and/or the RUL of an asset, without the requirement for in-depth knowledge of the physical nature of the asset and its degradation processes. A wide range of areas of study contribute to Data-driven modelling, many of which overlap with one another. Some of the most renowned of these fields are statistics, data mining and machine learning [4]. With Data-driven modelling, models are developed directly from data collected during condition monitoring exercises. The objective of Data-driven modelling is to correlate variations in an asset's condition monitoring data with the degradation of the asset and fault progression within the asset. The resultant model can be used to estimate RUL of the asset [34].

The process of correlating condition monitoring data with asset degradation is referred to as model training, which entails adjusting parameters of the model so that it accurately maps input data to output data. In the case of asset RUL estimation, the input data consists of historic

condition monitoring measurements and the output data consists of the actual RUL values associated with each of the input datum points. To build a Data-driven model that provides accurate physical asset failure prognoses requires a large data set consisting of the aforementioned historic input and output examples. Such data sets are referred to as training data sets. A large training data set is preferred because it is more likely to include a wide range of modes and phases of asset degradation as well as normal operating conditions of the asset. Therefore, such a data set can be used to train the model to make accurate predictions for a wide range of failure modes and operating conditions [34], [4].

The description of Data-driven modelling highlights the most notable advantages and disadvantages of this modelling approach. The key advantage of Data-driven modelling as opposed to Physics-based modelling, in the context of asset failure prognostics, is that there is no requirement for in-depth knowledge of the physics that describe the asset and its degradation processes. Furthermore, seeing as the model makes use of multivariate condition monitoring data to make prognoses, it is capable of making predictions that are generally more accurate than those made by Experience-based models that rely solely on time to failure data [34], [4].

The notable disadvantages of Data-driven modelling are that it requires a large set of input-output data in order for the model to be built and tested. Such data sets are difficult to acquire, especially for new assets that have not been monitored for long enough for a training set to be accumulated. Data-driven modelling also involves processes that require large amounts of computing power that might not be available. Finally, some Data-driven models are often criticised as being 'black-boxes', that make accurate predictions, but without clarifying the basis on which the predictions are made. Therefore, the reasoning process of the model is not clear, which reduces the amount of insights that can be extracted from the model. This disadvantage stands in contrast to the clarity with which Physics-based models are able to describe, in physics terms, the reasoning behind their predictions [34], [4].

3.2 Machine Learning

Machine learning (ML) is a concept that is difficult to define concisely due to the wide range of industries in which it is applied and the diverse set of disciplines to which ML is related. To illustrate, ML draws on ideas from statistics, computer science, artificial intelligence, data mining, neurobiology and psychology and has been applied in fields ranging from astronomy to particle physics [6], [5]. ML is an area of study primarily related to computer science, that aims to address the challenge of building computer programs that can learn to perform tasks based on experience, as opposed to conventional rule-based programming where computers perform

tasks based on ‘handmade’ and ‘hand-curated’ instructions that explicitly stipulate how the task should be performed [35], [36]. The concept of learning in the context of ML was broadly defined by T.M. Mitchell [36] as follows:

‘A computer program is said to learn from experience E with respect to some class of task T and performance P , if its performance at tasks in T , as measured by P , improves with experience E [36].

ML is especially useful for solving problems that are difficult to solve analytically but where there is enough example data that captures the essence of the problem that can be used to construct an empirical solution. Abu-Mustafa et-al. [18] illustrates this notion by using computer image recognition as an example. Consider writing a program that will allow a computer to recognise an image of a tree. It would be immensely difficult to represent the concept of a tree in code, and in such a way that a computer can use it to determine whether an image contains a tree or not. However, if we had a large data set of images that are labelled as either depicting a tree or not, we can feed them to a ML algorithm that will empirically develop a model that is capable of predicting whether an image contains a tree. The example relates to T.M. Mitchell’s [36] conceptualisation of learning in the context of ML. In the example, correctly classifying an image as depicting a tree or not is the task (T), the large data set of images of trees provides the experience (E), and if we were to present the resultant model with a set of new images, some of which depict trees and some do not, we can measure the model’s performance (P) as the percentage of images correctly classified by the model.

3.2.1 Types of ML

ML mainly concerns learning from data, where a set of observations is used to learn the underlying process that generated them. However, the immense breadth of this concept resulted in the formation of various subclasses of ML. ML model types are classified according to the nature of the data used to train the model. The most prevalent of these subclasses will be introduced in this section [18] .

3.2.1.1 Supervised learning

With Supervised learning an ML algorithm is provided with a data set that includes the target value associated with each of the observations that is in the set. The target values are referred to as output values and the data set is usually represented as a set of input-output combinations, $(\mathbf{x}_i, \mathbf{y}_i)$ [18], [37]. The subscript i indicates the index of the observation and therefore ranges from 1 to the number of observations in the training data set. The input and output variables are written in boldface to indicate that they are vectors, seeing as both these

values are often multidimensional. Supervised learning is referred to as such, because the learning algorithm is provided with true output values that act as a supervisor that monitors and guides the progress of the algorithm.

The task of the learning algorithm is to learn how to map the input values to output values. If we were in possession of every possible combination of the input space with their output values, no ML will be necessary seeing as the problem can be solved perfectly using a simple lookup function. However, such a data set is unlikely to exist for any interesting problems. Rather, the data sets that are used for Supervised learning contain a subset of the possible input space and their output values. The hope is that the employed algorithm will learn a mapping from the input to the output space that captures the underlying relationship between these two so that the resultant model will be able to make accurate predictions on input data that it has not encountered before.

3.2.1.2 Reinforcement learning

Reinforcement learning is when a learning algorithm is no longer provided with a data set containing the true output value for each of its input values, but rather with a score that informs the learning algorithm how good its prediction or decision was. With each iteration of the learning process the algorithm must try to adjust the way it makes decisions in such a way that it increases its score. The algorithm is often halted based on a predefined iteration count limit or by a minimum score improvement criteria of some sort. Reinforcement learning owes its name to the fact that it involves rewarding the learning algorithm with higher scores if it makes good decision and penalises the algorithm with low scores if it makes poor decisions and so the correct decision-making behaviour is reinforced [18] .

This approach is often taken when ML is used to teach a computer how to play a score-based game such as 'Tetris'. In the 'Tetris' example, the ML algorithm will control how descending shapes are moved and orientated. After each game, the accumulated score is fed back to the algorithm so that it can update its decision-making rules by which it plays the game. If the algorithm is built well and allowed to iterate over a sufficient number of games the resultant model should be able to play the game well.

3.2.1.3 Unsupervised learning

When a learning algorithm is provided with a data set containing only of input values it is referred to as Unsupervised learning. Such a data set is referred to as an unlabelled data set. The data in this scenario does not include the true output values associated with the input values, nor does the learning process include some form of feedback loop to reinforce good

predictions made by the model, that are related to and/or based on the input data. Yet, there is much that can be learnt from such a data set. Unsupervised learning is mainly concerned with finding patterns and structure in data and has become a key aspect of data mining. Data clustering and natural language processing are examples of Unsupervised learning techniques that are often used to extract information from unlabelled data sets. These two techniques can be used for discovering unknown market segments within a company's customer base and for identifying common themes in customer reviews, respectively [18], [38].

3.2.2 Classification versus regression

ML can also be classified according to the data type of the output value that the learning algorithm must learn to predict. When the output values are numerical the ML problem is referred to as a regression problem. Examples of regression problems include predicting a share price at some future time based on market related data or predicting a person's weight based on his or her demographic information [38].

When the output values are qualitative they can be aggregated into a finite number of classes. The task of the learning algorithm is then to produce a model that is capable of correctly predicting the class to which a set of input values belongs. Such problems are referred to as classification problems. Examples of classification problems include image recognition problems and predicting a person's ethnic group based on his or her demographic information. It is important to determine whether a ML problem is a regression or classification problem, seeing as some ML methods are better suited for one than for the other. For instance, linear regression is better suited for regression problems while logistic regression is better suited for classification problems. However, it is possible that linear regression might, purely coincidentally, show good results during model training for a classification problem, even though it might not be able to generalise well outside the training data set. Hence the importance of determining the ML problem type beforehand [38].

3.2.3 ML as a search

The remainder of this literature review will be focused on supervised classification ML problems. The decision to focus on supervised classification ML problems is based on the fact that the data provided by Metrorail, as described in Section 2.1.2, includes both input features as well as the accompanying output data. This makes the relevant case study of this investigation more suitable for supervised ML techniques, as opposed to cases where a data set does not include a target output variable, in which case unsupervised ML methods, such as clustering models, are more suitable. The terms 'model', 'learning algorithm' and 'input/output' values will now be

clarified in terms of supervised classification ML problems by providing a conceptual framework for such problems. The basic conceptual framework is illustrated in Figure 9 [18].

In Figure 9 [18] we see that there is an unknown function f that maps the input space \mathbf{X} to the output space \mathbf{Y} . Function f represents the real world system that is producing the output space \mathbf{Y} . The task is to approximate function f as closely as possible by using training examples $(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \dots, (\mathbf{x}_N, \mathbf{y}_N)$ that have been produced by f . This is done by postulating a model type that will be able to fit the training data well. This postulation defines the hypothesis set \mathbf{H} . If it seems as though there is a linear relationship between \mathbf{x} and \mathbf{y} , for example, then \mathbf{H} will consist of all possible linear combinations of the input space \mathbf{x} .

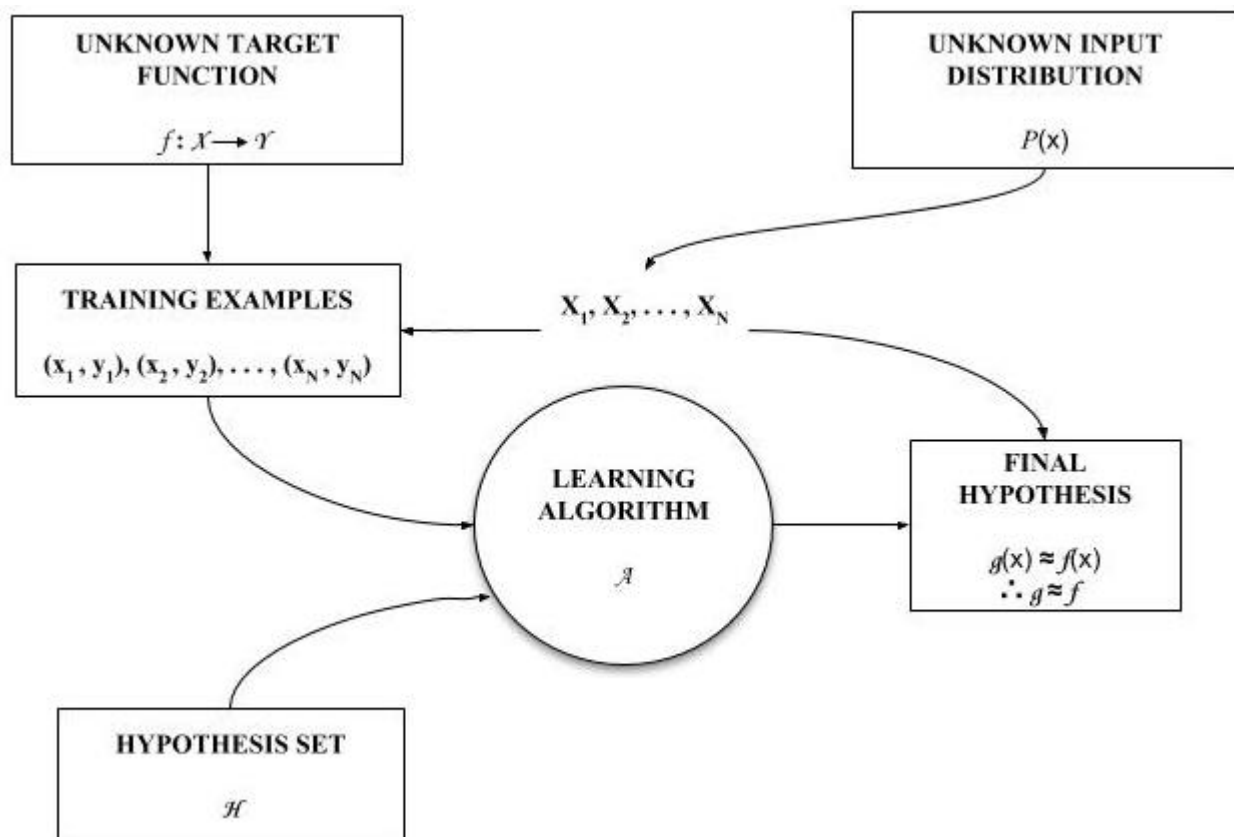


Figure 9: Conceptual framework of Supervised ML problem, adapted from [18]

The hypothesis set and the training examples are passed to the learning algorithm \mathcal{A} . The learning algorithm searches through \mathbf{H} until it finds an element of \mathbf{H} , namely g , that approximates f the best on the training data, based on some performance measure. An example of a learning algorithm would be least squares regression for the aforementioned linear model example. The hope is that g will continue to approximate f well for data that are not in the training examples. The reason why we cannot be one hundred percent certain that the model

will always perform as well as it did on the training data is due to the unknown probability function $P(\mathbf{x})$. Note that in Figure 9, the composition of the available sample of training examples is dictated by f and an unknown probability distribution $P(\mathbf{x})$. $P(\mathbf{x})$ is included to emphasise the fact that our training set does not include all possible \mathbf{x} and \mathbf{y} combinations and that we only received a specific training set by chance, albeit to a lesser or greater extent. The selected hypothesis, g , is referred to as the final model that is produced by the ML process [18].

3.2.4 ML models

A conceptual framework of supervised classification ML problems was presented in the previous section. This section builds on this framework by introducing some of the most common supervised classification ML models and learning algorithms in use today.

3.2.4.1 Logistic Regression

In many supervised classification ML problems, the output value to be modelled takes the form of a binary variable. Despite the simplicity and interpretability of linear regression, it is not suitable for binary classification problems. One of the reasons for this is that a linear regression model is capable of producing predictions ranging from negative to positive infinity even though the target variable can only take on one of two values [38].

The inception of Generalised Linear Models (GLM) led to modelling techniques that are analogues to linear regression, that are suitable for classification problems. Logistic regression is a GLM that is especially well suited for binary classification problems. In order to explain the logistic regression model and its learning algorithm, some model elements and notation need to be defined [38].

Let Z be a vector consisting of all M observations of the independently generated, binomial random output variable. Each element of Z is either 1 or 0, the interpretation of which is dictated by the specific case. Let X^* be a $M \times (K+1)$ matrix consisting of all M observations of the K input variables and one column consisting of 1's. For convenience we create X , an $N \times (K+1)$ matrix, by aggregating X^* so that each of the N rows is a unique combination of the observed $K+1$ input variables. Each element of X is referred to as a population, where x_i is the i^{th} population and $i \in 1, 2, \dots, N$. Let N be an $N \times 1$ vector, of which each element n_i is the number of observations in population i . It therefore follows that $\sum_{i=1}^N n_i = M$. Now, let Y and p be $N \times 1$ vectors, where y_i is the number of successes in population i and p_i is the probability of success for an observation in population i , i.e. $p_i = P(Z_i = 1 | i)$. Finally, let β be a $(K+1) \times 1$ vector consisting of the model parameters, where each element β_k is associated with one of the $K+1$ input variables.

The logistic regression model sets the logit (log-odds) of the probability of success for population i equal to a linear combination of the model parameters β and inputs x_i . Its generic shape is illustrated in Figure 10 and the function is expressed in Eq.1.2.

$$p_i = \frac{e^{\sum_{k=0}^K x_{ik}\beta_k}}{1 + e^{\sum_{k=0}^K x_{ik}\beta_k}} \quad (1.1)$$

$$\log\left(\frac{p_i}{1 - p_i}\right) = \sum_{k=0}^K x_{ik}\beta_k \quad (1.2)$$

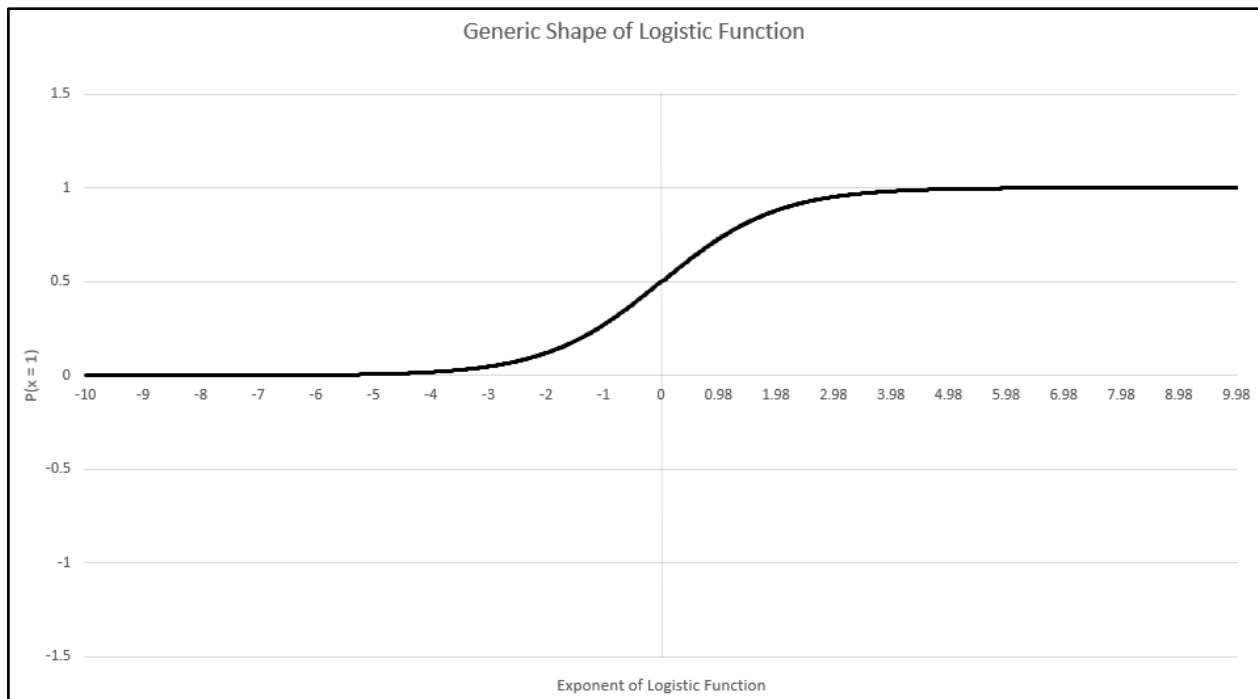


Figure 10: Generic shape of the logistic function (Eq. 1.2)

Maximum Likelihood Estimation (MLE) provides a mathematical approach to finding an optimal set of values for the vector of model parameters, β . MLE seeks to select elements for β such that the probability of realisation of the observed output values is maximised. If we assume that y_i is a binomial count for population i , then the likelihood function to be maximised is:

$$L(\beta|y) = L(\beta) = \prod_{i=1}^N \frac{n_i!}{y_i!(n_i - y_i)!} p_i^{y_i} (1 - p_i)^{n_i - y_i} \quad (2)$$

After solving for p_i in Eq. 1, substituting p_i in Eq. 2 and some further simplification we find the log likelihood form of Eq. 2 (Note the constant $\prod_{i=1}^N \frac{n_i}{y_i}$ is omitted during this process for simplicity), provided in Eq. 3.

$$l(\beta) = \log(L(\beta)) \propto \sum_{i=1}^N y_i \left(\sum_{k=0}^K x_{ik} \beta_k \right) - n_i \left(\log \left(1 + e^{\sum_{k=0}^K x_{ik} \beta_k} \right) \right) \quad (3)$$

In the first step to finding the solution to $\arg \max_{\beta} l(\beta)$, we set each of the $k + 1$ partial derivatives $\frac{\partial l(\beta)}{\partial \beta_k}$ equal to zero and solve for each β_k . This gives the critical points of Eq. 3, which is either a maximum or minimum point. Then, to insure that the solved values specify a maximum point we find the second order partial derivatives of Eq. 3 with respect to each of the β_k variables and insure that the values for each of the $k + 1$ equations is negative when we substitute β_k with the values derived in the first step. The Newton-Raphson method provides the means for finding the critical points of Eq. 3. This method follows the following algorithm [39] :

Algorithm 1: Newton-Raphson Method for Finding the Roots of $\frac{\partial l(\beta)}{\partial \beta_k}$

Let:

$$f(\beta) = \frac{\partial l(\beta)}{\partial \beta_k};$$

$$f'(\beta) = \frac{\partial^2 l(\beta)}{\partial \beta_k \partial \beta_{k'}} \text{ (i.e. the second order partial derivatives of Eq. 3 with respect to each } \beta_k \text{);}$$

Initiate:

Generate $\beta^{(j)} = \beta_k^{(j)}$ for $j = 0$ by guessing initial values for each β_k ;

Repeat Until Convergence over $j = 0, 1, 2, \dots$:

$$\text{Set } \beta^{(j+1)} = \beta^{(j)} + f(\beta^{(j)})[f'(\beta^{(j)})]^{-1};$$

Making class predictions is relatively simple once the coefficient β is estimated. Given an input observation, \mathbf{x} , the predicted probability that $y = 1$ for the observation is calculated using Eq. 1.1. The final step is to decide on a threshold value for $P(y = 1|x)$, above which the output is predicted as 1. A popular choice is 0.5, i.e. the prediction is $y = 1$ if $P(y = 1|x) \geq 0.5$, owing to the fact that a threshold of 0.5 is an unbiased threshold which provides a starting point for testing model performance.

3.2.4.2 Artificial Neural Networks

An Artificial Neural Network (ANN) is a biologically inspired ML model that mimics the way in which the brain processes information. The brain contains numerous basic information processing cells called neurons. Each neuron is connected to a large number of other neurons by synapses and axons. In very basic terms, a neuron receives information in the form of impulses, from other neurons via synapses. The impulses are combined in the neuron and if the combination of impulses excites/activates a neuron, the neuron in turn sends an impulse to its subsequently connected neurons through its axon [40], [41]. This structure is illustrated in Figure 11 [40] .

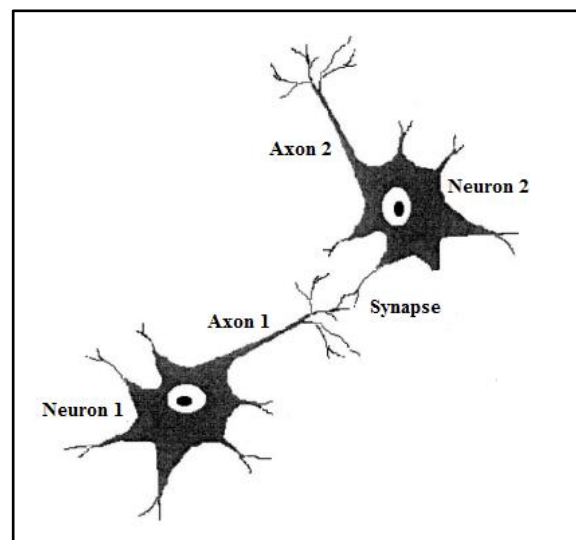


Figure 11: Illustration of two connected neurons, Adapted from [40]

In 1943 a mathematical model of how neurons process information was proposed, named McCulloch-Pitts Neurons [37]. The model is illustrated in Figure 12 [37]. In Figure 12 we see a single neuron connected to M other neurons that supply binary inputs $x_1, x_2, x_3, \dots, x_M$. The strength of an input is determined by the conductivity of the connecting synapse, which is represented by weights $w_1, w_2, w_3, \dots, w_M$. The inputs multiplied by the weights are aggregated by the neuron, the result is passed to a step function, referred to as the activation function, with threshold θ . If the result is greater than θ the neuron fires, producing an output of 1, alternatively it produces an output of 0 [37].

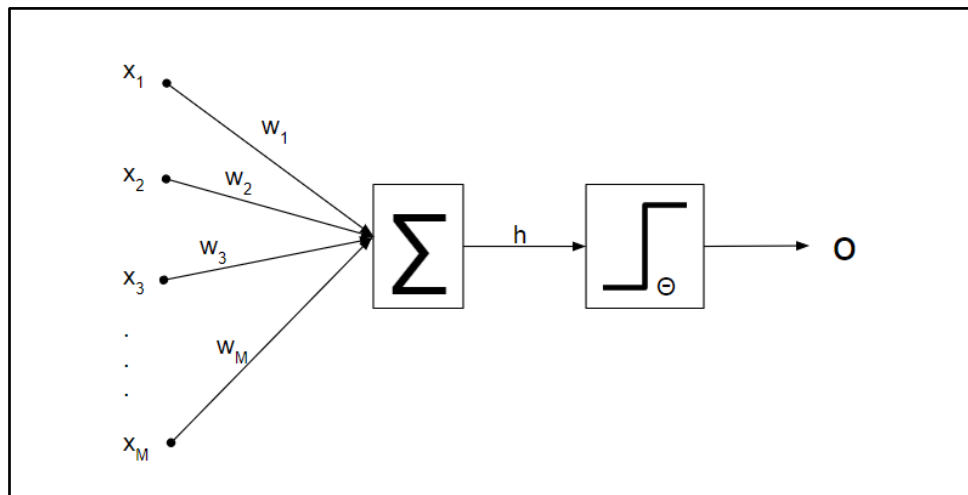


Figure 12: McCulloch-Pitts neuron model, adapted from [37]

Clearly a single neuron like the one illustrated in Figure 12 is not capable of modelling complex functions. It can be shown that a series of such neurons connected in parallel is only capable of modelling discriminant linear lines, planes or hyperplanes, depending on the number of neurons, severely limiting their utility. However, ANNs that are capable of modelling complex functions are created by connecting layers of neurons together, in other words, the output of a layer of neurons serves as the input of a subsequent layer of neurons. In this way, the input variables are transformed into more and more complex features that the neural network can use to make complex decisions. In fact, the universal approximation theorem states that an ANN with a single hidden layer containing a finite number of hidden neurons can approximate any continuous function perfectly [37].

The architecture of an ANN, as well as the choice of activation function and training algorithm plays a crucial role in determining its performance [41]. An inexhaustible combination of these configurations has been proposed in the literature. Therefore, this review will focus on a very popular ANN schema, namely, a fully connected, feedforward ANN with one input layer, one hidden layer and one output layer. The input and hidden layers will each have one bias neuron. The activation function of the hidden and output layer neurons will be the logistic function (Eq. 1.1) and the back propagation algorithm will be used to train the model. Such an ANN is illustrated in Figure 13. This ANN schema was selected due to the fact that various authors in the field of ML refer to it when discussing ANNs for classification purposes, which is particularly relevant to the focal case study of this project [37], [38], [40], [41]. Furthermore, this relatively simple ANN schema includes the various elements of an ANN model, which means that its selection for discussion does not limit the detail with which this model type can be discussed.

An ANN model is trained by adjusting the weights connecting the neurons. The challenge is to determine what effect a change in one of the weights in the first layer will have on the prediction of the output nodes, moreover, how the weights must be adjusted to improve the accuracy of the model's predictions. The back propagation algorithm provides a means by which the derivative of the error of prediction can be obtained with respect to any weight in the network. Gradient descent can then be used to adjust the weights in a way that will reduce the error, once this derivative is calculated.

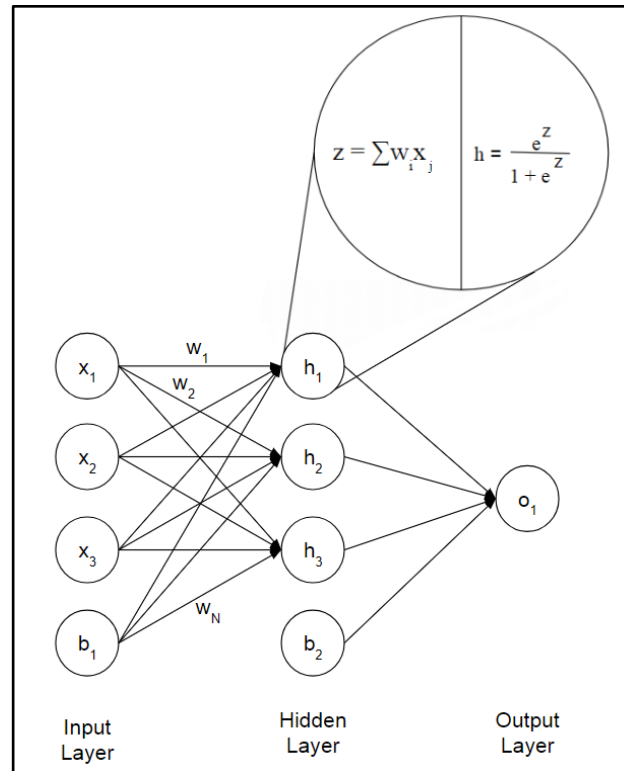


Figure 13: Feedforward ANN

Some notation must be clarified to describe how an ANN like the one depicted in Figure 13 generates predictions for input values, and how the model is trained using the back propagation algorithm. Consider an $N \times M$ matrix, \mathbf{x} , which is comprised of N input observations of $M-1$ features and one bias feature with value 1. Then, let \mathbf{y} be a vector of length N , comprised of observed binary output values related to each of the N inputs. The dimensions of the network are as follows: The input layer consists of M nodes, one for each of the M input features. The hidden layer contains H nodes and one bias node. The output layer consists of a single node, which produces a value between 0 and 1, which is rounded off to produce a binary output. Let $\mathbf{w}^{(1)}$ be the $M \times H$ weight matrix connecting the input layer to the hidden layer and let $\mathbf{w}^{(2)}$ be the $(H+1) \times 1$ weight matrix connecting the complete hidden layer to the output layer. Both weight vectors are initiated with guessed values.

The ANN produces predictions according to the following algorithm:

Algorithm 2: ANN Prediction Generation

Compute:

$$\mathbf{z}^{(1)} = \mathbf{x} \cdot \mathbf{w}^{(1)}$$

$$\mathbf{o}^{(1)} = \frac{e^{\mathbf{z}^{(1)}}}{1 + e^{\mathbf{z}^{(1)}}}$$

$$\mathbf{z}^{(2)} = \mathbf{o}^{(1)*} \cdot \mathbf{w}^{(2)}$$

$$\mathbf{o}^{(2)} = \frac{e^{\mathbf{z}^{(2)}}}{1 + e^{\mathbf{z}^{(2)}}}$$

Round for Final Predictions:

$$\mathbf{p} = \lfloor \mathbf{o}^{(2)} \rfloor$$

Note:

- 1) Superscripts (1) and (2) refer to the hidden layer and the output layer respectively.
- 2) $\mathbf{o}^{(1)*}$ consists of $\mathbf{o}^{(1)}$ with a column of ones appended to it to accommodate the bias node in the hidden layer.

The back propagation algorithm utilises the chain rule to relate changes in any weight in the network to changes in the prediction error of the model. The exact process is discussed in detail in [36], [37] and [41], therefore only the final update steps of the algorithm will be shown here. A popular measure of prediction error for back propagation implementation is the Mean Squared Error (MSE) which is expressed as [37], [40] :

$$MSE = \frac{1}{2N} \sum_{i=1}^N \left(y_i - o_i^{(2)} \right)^2 \quad (4)$$

Using the notation introduced in Algorithm 2, the weights in $\mathbf{w}^{(2)}$ are updated as follows:

Let:

$$\xi_i = \left(y_i - o_i^{(2)} \right) \cdot o_i^{(2)} \cdot \left(1 - o_i^{(2)} \right) \quad (5)$$

Where i ranges from 1 to N.

Then:

$$\Delta \mathbf{w}^{(2)} = \frac{-\alpha}{N} \begin{bmatrix} \sum_{i=1}^N \xi_i \cdot o_{i,1}^{(1)} \\ \sum_{i=1}^N \xi_i \cdot o_{i,2}^{(1)} \\ \vdots \\ \sum_{i=1}^N \xi_i \cdot o_{i,(H+1)}^{(1)} \end{bmatrix} \quad (6)$$

The update matrix for $\mathbf{w}^{(1)}$ is calculated as:

$$\Delta \mathbf{w}^{(1)} = \frac{-\alpha}{N} \sum_{i=1}^N \xi_i \cdot \mathbf{x}_i \cdot [\mathbf{o}_i^{(1)} \cdot (1 - \mathbf{o}_i^{(1)})^T \cdot \mathbf{w}^{(2)}]^T \quad (7)$$

In both Eq. 6 and 7, α is a scaling factor, usually set to $0 < \alpha \leq 0.5$. This parameter is the learning rate, which is intended to reduce the step size for each iteration of the back propagation algorithm. The reduction in step size improves the stability of the training algorithm and helps to insure that the back propagation algorithm converges [37].

The ANN weights are updated with the results from Eq. 6 and 7 as follows:

$$\mathbf{w}_{\text{new}}^{(1)} = \mathbf{w}_{\text{old}}^{(1)} - \Delta \mathbf{w}^{(1)} \quad (8.1)$$

$$\mathbf{w}_{\text{new}}^{(2)} = \mathbf{w}_{\text{old}}^{(2)} - \Delta \mathbf{w}^{(2)} \quad (8.2)$$

These updates are made to the weight matrices until a stoppage criterion is reached. This criterion is often defined as a fixed number of iterations or by stating a minimum change in the output error below which the algorithm is assumed to have converged [37]. Once the stoppage criterion is satisfied the model is trained and Algorithm 2 can be used to make predictions on unseen input data.

The discussion of ANNs will be concluded with some practical notes on its implementation. Four key aspects that needs to be considered is:

- 1) How should the network weights be initialised?
- 2) How will the risk of back propagation converging to a local minimum be reduced?
- 3) How many hidden layers should there be in the network?
- 4) How many neurons should there be in each hidden layer?

The first two points can be addressed simultaneously. Generally, it is advantageous to initialise weights of an ANN to values close to zero if the network makes use of a logistic activation function. This allows gradient descent to adjust the weights of the network at a fast rate, initially, which is equivalent to the model learning quickly during the early stages of training. The reason for this is due to the fact that the derivative of the logistic function is large when the function's exponent is close to zero, as seen in Figure 10 [36], [37].

Literature suggests that, if the input space is made up of N variables that have been normalised, the network weights should be initialised to a set of values that are uniformly distributed between $-\frac{1}{\sqrt{N}}$ and $\frac{1}{\sqrt{N}}$ [37]. It is also suggested that a number of different ANNs should be generated with unique, randomly generated sets of weights, and that each network should be put through a training procedure. The ANN that provides the best performance based on a validation process will then be selected as the final model.

This process also addresses point two, seeing as selecting a winning network from the generated ANNs will reduce the risk of selecting a model that converged to a poorly performing local minimum [37]. Another technique for escaping local minima is to add 'momentum' terms, $\eta \cdot (\Delta w^{(1)}_{t-1})$ and $\eta \cdot (\Delta w^{(2)}_{t-1})$ to Eq. 8.1 and 8.2, respectively. η is the momentum coefficient, the value of which is often initialised to between zero and one and adjusted through trial and error, while $(\Delta w^{(1)}_{t-1})$ refers to the weight update for the previous back propagation iteration.

The idea is that this term will help by prohibiting the weight updates from drastic changes in direction and by nudging the training process out if it gets stuck in a local minimum, similar to the effect that added momentum will have on a rolling ball [36], [37], [41].

The number of hidden layers to include in an ANN and the number of nodes that these layers should contain are generally considered as tuning parameters, forming part of the model training process. This is because the optimal decision of these model design parameters is practically impossible to know at the onset of the modelling process. Therefore, as with weight initiation, points 3 and 4 of the above list can be addressed by generating a number of ANNs with different architectures and selecting a best performing network. There are, however, numerous informal guidelines, relating a prescribed number of nodes in the hidden layer to the dimensions of the input and output space of the problem. Basheer and Hajmeer list some of these rules in [41], though they advise that trial and error should be used to select a good performing architecture. It is also advised that one should start off with a small set of nodes in the hidden layer and grow the network out until adequate performance is achieved. Finally, it is generally advised that no

more than two hidden layers should be used, seeing as both one and two hidden-layer networks can already model complex behaviour and adding more hidden layers will produce a model that will generalise poorly [37], [41].

3.2.4.3 Random Forest

Random forest is an ML modelling technique that provides a means to improve the predictive performance of tree-based models. Tree-based models are supervised ML models that work by sequentially segmenting the input space of a problem into a number of regions. A classification prediction is then made by determining which region an input belongs to and then returning the mode output associated with that region. In the binary output case, segmentation is done by taking an input variable and finding a threshold value for that variable that splits the output data into sets that exhibit the highest possible degree of purity. The threshold is referred to as a decision boundary, therefore these models are also referred to as decision trees. This splitting process continues sequentially and for each split a different input variable can be selected to create a decision boundary. The process continues until regions are produced that contain a predefined minimum number of observations or level of region purity, where purity refers to the degree to which a region contains only a specific class of output variable [37], [38]. Decision trees get their name from the fact that a visual representation of the sequential decision boundaries and terminal regions, referred to as leaf nodes, resembles a tree-like structure, as seen in Figure 8 [38].

A popular measure for leaf node impurity in the binary classification case is the Gini Impurity score [37], [38]. This score is calculated, the splits made with a given input variable, as follows:

$$G_i = \sum_{m=1}^M p_m \cdot p_{m,k} \cdot (1 - p_{m,k}) \quad (9)$$

where G_i is the Gini Impurity score for variable i , M is the number of classes (i.e. splits) of variable i , p_m is the proportion of observations that belong to class m of variable i and $p_{m,k}$ is the proportion of observations that belong to class m of variable i and class k of the binary output variable. The variable with the lowest Gini Impurity score is selected to make the next split in the decision tree [37], [38].

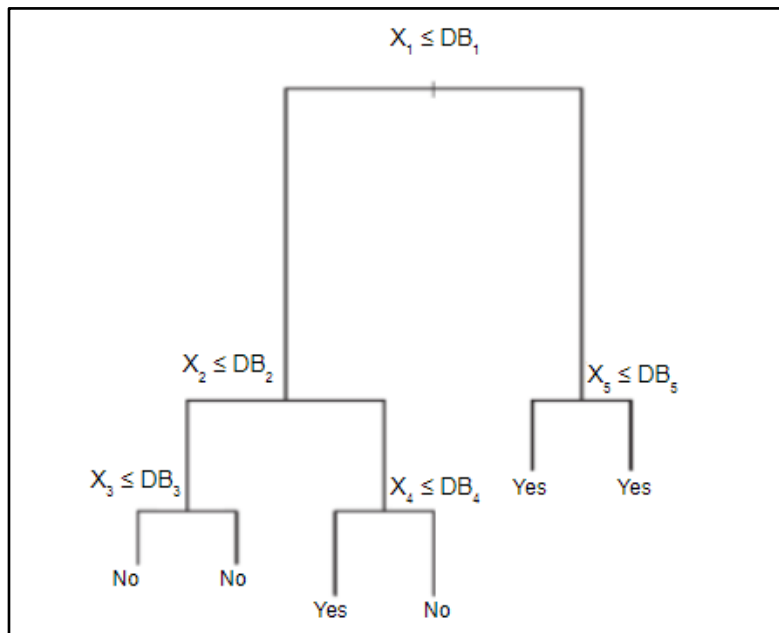


Figure 14: Visual representation of decision tree, adapted from [38]

Some of the advantages of decision trees are that they are extremely simple, making them easy to implement and easy to interpret. Once built, they also allow for an easily interpreted visual representation, as seen in Figure 14. Finally, decision trees are capable of handling quantitative and qualitative input variables with the same level of ease. However, an important disadvantage of decision trees is that it is often necessary to implement greedy splitting rules like the Gini Impurity score, especially if there are a large number of input variables, which could lead to a poorly performing model [38].

Decision trees in general tend to exhibit high variance, which means that they are highly sensitive to changes in the training data. For instance, if one were to divide a training data set in half and build a decision tree for each half, these decision trees could look vastly different. A model with high variance is prone to overtraining, a phenomenon that leads to poor model generalisation. Random forest involves generating a large number of decision trees, each on a randomly resampled subset of the training data. Then, by selecting the mode prediction over all the generated trees as the final prediction, the high variance of a single decision tree is mitigated. This occurs due to the fact that the variance will be inversely proportional to the number of generated decision trees. This process of generating a large set of models to reduce prediction variance is known as bagging. However, further steps need to be taken for decision trees, due to the fact that the presence of a small set of variables that are able to create very pure leaf nodes will always make up the first set of splits for all the generated trees. This results in generating highly correlated trees, which does not address the high variance issue of a single decision tree. Therefore, with a random forest, each decision tree is not only presented with a

randomly sampled subset of the observations, but also with a random subset of input variables. This forces the decision trees to be less correlated, resulting in improved generalisation. A disadvantage of using random forest instead of a decision tree is that the simplicity and interpretability of a decision tree is sacrificed for better predictive performance [38], [42].

The random forest procedure works as follows: Let Z be the $N \times M$ matrix consisting of N observations of M input variables. Let Y be the $N \times 1$ vector of a corresponding binary output variable. We then generate B sets of $\hat{Z}^{(i)}$, with i ranging from one to B , by randomly sampling K observations, with replacement, of L randomly selected variables from Z . The dimensions for each $\hat{Z}^{(i)}$ is therefore $K \times L$ where K and L are set to a values $K \leq N$ and $L < M$.

For each $\hat{Z}^{(i)}$ we build a decision tree, using the Gini Impurity score with regards to $Y^{(i)}$ to determine the splitting sequence of the L input variables. The splitting procedure continues until a minimum leaf node size, n , is reached for all leaf nodes. Classification predictions are then made by choosing the mode prediction made by the B decision trees. The values of B , K , L and n are regarded as model tuning parameters. Therefore, random forests are generated for a number of candidate configurations of these parameters, and the best performing configuration is selected [42].

One of the attractive aspects of the random forest model is that the risk of overfitting the training data does not increase with an increase in the parameter B . An increase in B only results in a reduction in the variance of the mode predictions made by the generated decision trees. Therefore, this parameter is easy to configure, seeing as one can start with a small value for B and increase it until the prediction accuracy of the model stabilises, without having to worry about reducing the generalisability of the model. Another attractive aspect of random forest is the concept of Out-Of-Bag (OOB) error. For each decision tree generated by random forest, the OOB observations are those that were not contained in the decision tree's K observations. For each observation in N , we can determine for which of the B decision trees in the model it was OOB. These decision trees can then be tasked with classifying this observation as a test. After doing so for each of the N observations the overall misclassification rate is the OOB error rate. This measurement can be used to tune the aforementioned model parameter configurations. The advantage lies in the fact that, with OOB error, we do not require a hold-out data set to test the model parameter configurations, which allows random forest to consume data very economically [38], [43].

3.2.5 Hyperparameter tuning and model evaluation

In the previous subsections two popular ML models, namely ANN and random forest, were introduced and their training algorithms, or at least a variant of their training algorithms, were described. Both models had some form of a tuning parameter, usually referred to as hyperparameters, which are not optimised by the model's training algorithm, even though they influence the predictive performance of the final model. The hyperparameters referred to in the discussion of the these ML models are listed in Table 3.

In general, there is no easy way of estimating optimum values for these parameters. There are rules of thumb, established through experience, that suggest ranges for some of these parameters; however, the final values are generally selected based on trial and error testing. Grid-search, and k-fold cross-validation are two techniques that are used to automate this trial and error process.

Table 3: List of discussed ML model hyperparameters

| Parameter No. | Model | Hyperparameter |
|---------------|---------------|--|
| 1 | ANN | Number of hidden layers in the network |
| 2 | ANN | Number of neurons in the hidden layer(s) |
| 3 | ANN | Value of Momentum coefficient (η) |
| 4 | ANN | Value of learning rate (α) |
| 5 | Random Forest | Value of minimum leaf node size (n) |
| 6 | Random Forest | Number of decision trees to generate (B) |
| 7 | Random Forest | Number of observations to sample per decision tree (K) |
| 8 | Random Forest | Number of variables to sample per decision tree (L) |

3.2.5.1 Grid-search

Grid-search is an algorithm that entails searching through all combinations of a set of parameters and determining which combination produced the best performing model based on

some performance measure. Grid-search requires explicit expression of the algorithm's searchable space, which means that each of the parameters must be bounded and all addressable values for each parameter must be stated. This produces a grid of values which the algorithm can traverse in search of a combination that produces optimal model performance, hence the name grid-search [44]. This delimited and discretised parameter search space is illustrated in Figure 15 [45]. Figure 15 illustrates a model's performance w.r.t. a grid formed by combinations of its two hyperparameters.

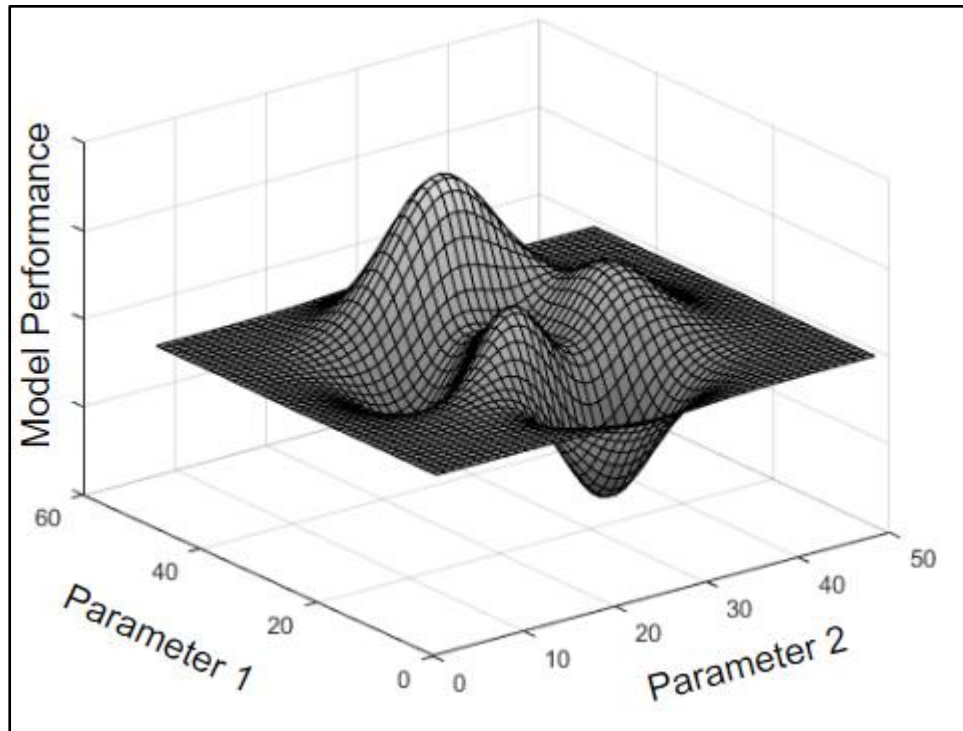


Figure 15: Visual representation of Grid-search surface with respect to model performance, adapted from [45]

3.2.5.2 ML Model Performance Evaluation

Measuring the predictive performance of a model is a crucial step in ML. This measurement is used to compare candidate models and to establish the accuracy that one can expect from a deployed model. Both functions are immensely important to the modeller. The first function is used as a baseline measurement which indicates when a model's accuracy is deteriorating, which could mean that the underlying process producing observations has changed. This would call for model revision and redevelopment. The second function is important for model development, and especially hyperparameter tuning. This is due to the fact that the realisation of a model with differing hyperparameters can be thought of as two candidate models which ought to be compared, based on some performance measurement, so that the best model can be

deployed. This performance measurement is emphasised in Figure 15, where each intersection of the two hyperparameters constitutes a different model that exhibits its own level of performance.

The confusion matrix is a tool for measuring the performance of a classification model. It entails constructing a square matrix with rows and columns indexed by the output variable classes. Each column represents a class predicted by the model and each row represents an actual observation class, therefore a confusion matrix of a binary output variable would be a 2x2 matrix. Each element of the matrix is comprised of the frequency of predicted versus actual occurrences [37]. The layout of a binary confusion matrix and its constituents is illustrated in Table 4.

Table 4: Confusion Matrix Layout

| | Predicted Class = 1 | Predicted Class = 0 |
|-------------------------|-------------------------------|-------------------------------|
| Actual Class = 1 | True Positive Frequency (TP) | False Negative Frequency (FN) |
| Actual Class = 0 | False Positive Frequency (FP) | True Negative Frequency (TN) |

The important metrics that can be derived from such a confusion matrix are sensitivity, specificity, accuracy and the F_1 score. These metrics are calculated as follows:

$$Sensitivity = \frac{TP}{TP + FN} \quad (10)$$

$$Specificity = \frac{TN}{TN + FP} \quad (11)$$

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \quad (12)$$

$$F_1 = \frac{TP}{TP + (FN + FP) / 2} \quad (13)$$

The reason why Eq. 12 alone is not a sufficient measurement of model performance is because of output variable class imbalance. For instance, if ninety percent of output variable observations fall in one class, then model accuracy would be 90% by simply predicting that every observation belongs to that class. This would seem like good predictive performance, even though the model is obviously flawed. Sensitivity and specificity provides more detailed measurements for model

performance. In the binary classification case, such as the one depicted in Table 4, specificity measures a model's ability to classify positive output observations by calculating the proportion of actual positive observations that the model was able to classify correctly. Specificity is the same measurement, but only for the negative output variable class. The F_1 score is similar to accuracy seeing as it provides a single score for a model's performance; however, an F_1 score provides a more balanced score than pure accuracy by taking into account a model's ability to classify positive as well as negative output observations, instead of just measuring the proportion of correctly classified observations [37].

Confusion matrices are useful model performance measurement tools, especially in cases where there is a big difference between the cost associated with a type I and a type II prediction error. However, it is also desirable to have a means of measuring a model's performance so that the measurement is not affected by an imbalance in the class frequencies of the output variable. Furthermore, confusion matrices only provide an indication of the performance of a classification model for a specific decision threshold. The Receiver Operating Characteristic (ROC) curve is a performance measurement tool that extends the functionality of decision matrices. Some of the desirable characteristics of ROCs are that the measurements produced by them are unaffected by output variable class proportions, they provide an easily interpretable visual representation of classifier performance, and finally, they evaluate classifier performance over a range of classification decision thresholds and produce a single summary measurement known as Area Under Curve (AUC) [37], [46].

An example of an ROC curve is illustrated in Figure 16. ROC curves are produced as follows: A trained binary classification model is tasked with classifying input observations of a test data set. The classification decision threshold is slowly increased from zero to one and for each value the sensitivity and specificity of the model is calculated based on the predicted versus actual classes of the test data set output variable.

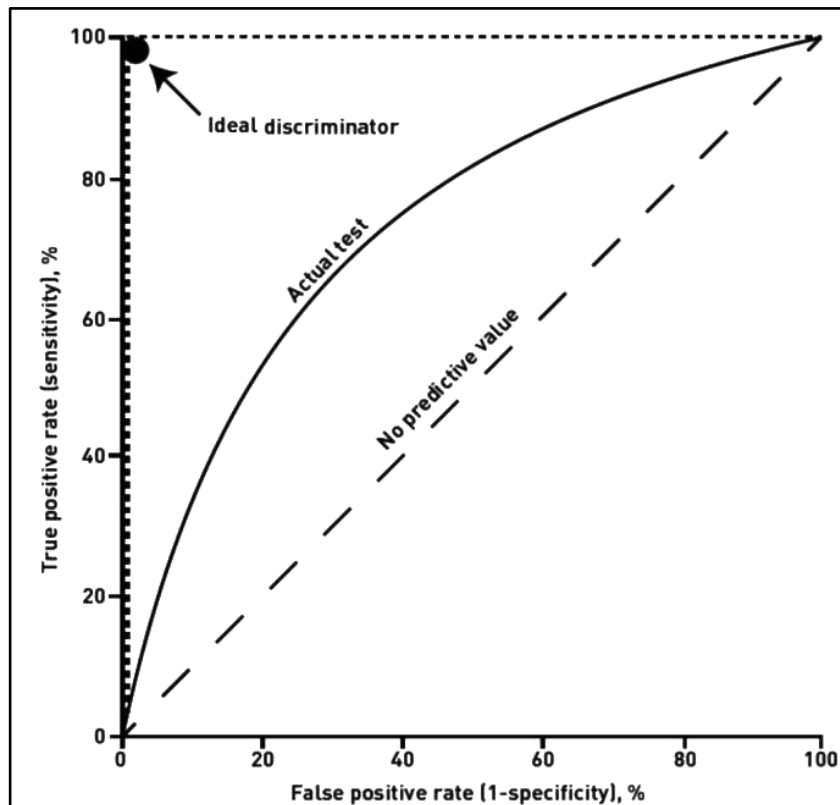


Figure 16: Example of an ROC curve, adapted from [47]

For each decision threshold value, the sensitivity is plotted against the 1-specificity as shown in Figure 16 [47]. ROC curves usually include a reference line, indicated by the dashed line, that stretches from the origin at a positive 45° angle. This line represents a worst case scenario, where the model has learnt nothing and exhibits predictive performance that is no better than 50/50 chance [37], [48].

ROC curves can be interpreted by first considering the effect of setting the decision threshold of a model to the two extreme cases. If the decision threshold is set to zero, then the model predicts true/1 for all observations, in which case the true positive rate will be 100%, but the cost is that the false positive rate will also be 100%. Setting the decision threshold to one forces the model to predict false/0 for all observations, resulting in both a false positive and true positive rate of 0%. An ideal model that separates the output variable classes perfectly will produce an ROC curve that hugs the Y-axis, as the decision threshold is decreased from one, until the point is reached where the true positive rate is 100% while the false positive rate is 0%. Only after this point will a further decrease in the decision threshold start to increase the false positive rate. This ideal discriminator is indicated in Figure 16 by the dotted line. From this interpretation it is clear that the closer an ROC curve gets to the top left corner of the chart area, the better it performs, however, if a modeller were to rely on a single point on the ROC curve to evaluate a

model's performance, then a sequentially generated set of confusion matrices would have provided the same utility. AUC is a performance measurement that provides a better indication of overall model performance and it is defined as the area that is under the ROC curve. In the perfect case AUC is equal to one and in the worst case AUC is equal to 0.5. AUC is a better measurement of model performance than a single point on the ROC curve because it conveys the closeness of an ROC curve to the top left corner of the chart area as well as the degree to which a model's behaviour adheres to what is expected, in one metric. The former is a consequence of how AUC increases as the shortest distance between the ROC curve and the top left corner of the chart area decreases. The latter is explained as follows: The expectation is that a classification model will change from being overly conservative to overly lenient as the decision threshold is decreased from zero to one. If we hold all else constant, a model that exhibits this behaviour will produce a larger AUC than one that does not.

3.2.5.3 K-Fold cross-validation

Model validation is performed so as to ascertain how well a model is able to generalise, i.e. produce accurate predictions on real-life observations that did not necessarily occur in the model's training data. As mentioned previously, this is one of the foremost goals of ML. The simplest approach to model validation would be to deploy a trained ML model and measure the predictive performance of the model. However, it would be advantageous to be able to determine how well a model generalises before deploying it, especially in cases where poor model performance can incur large costs to a company or put people's safety at risk.

One method of doing so is to simulate model deployment by presenting a model with a data set that it has not encountered yet. This method entails randomly splitting a data set into a train and test data set according to predefined proportions, e.g. 60% of the data set is randomly sampled without replacement to create a training set and the remainder forms the test set. The model is then trained on the training set and its ability to generalise is measured based on the prediction performance on the test set. This method also helps to determine when a model is starting to become overtrained. This is when a model is trained to the point where it becomes overly specialised at predicting observations that are in its training data set and is no longer able to generalise outside of that set [38]. Figure 17 illustrates typical training set vs. test set model performance, based on MSE, over a set of iterations of its training algorithm.

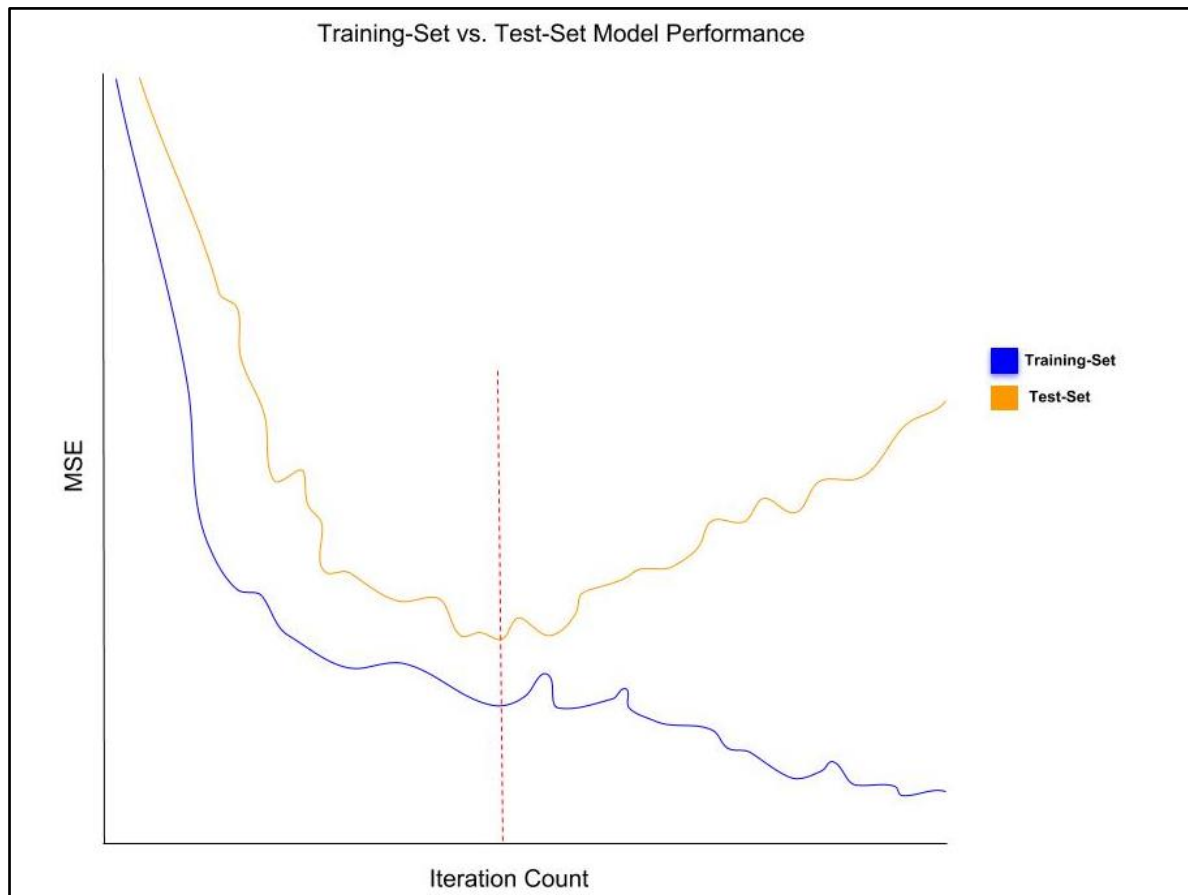


Figure 17: Illustration of model performance measured on training set (blue) vs. test set (orange) with ideal training stoppage point indicated (red)

In Figure 17, MSE for the training set has a decreasing trend that is sustained as the number of training algorithm iterations increase. For some ML models it is possible to reduce this MSE to zero; however, when we look at the MSE for predictions made for test set observations, depicted by the orange curve, we see that this feat is of little value to the modeller. The test set MSE decreases with the training set MSE, initially, during the period where the model is narrowing down on the actual function that is producing the data of which both these sets are comprised. This continues up until the point where the model starts to learn the features of the training set that makes it unique, at which point the noise in the training set is being modelled instead of the aforementioned function. It is from this point forward that the model starts to lose its ability to generalise, which is indicated by the reversed trend of the MSE rate of the orange curve. The validation set, therefore, indicates to the modeller when the training process should be halted, which is roughly indicated by the red dashed line.

This 'hold-out set' method introduces two problems that are both related to the fact that this method resorts to the usage of a single, unique test set. Firstly, the fact that, like the training set,

the test set may contain observations that make it unique, makes this method vulnerable to test set variability. In other words, Figure 17 might look very different, based on the observations that were randomly selected to create the training and test sets, which reduces the modeller's confidence in the outcome of the validation process. Secondly, this method is ill-suited when a model includes hyperparameters that need to be tuned. In fact, an additional validation set is required for each aspect of a modelling process that involves optimisation or comparison.

This notion can be explained in terms of competing models and the aforementioned concept of overtraining. Consider the case where three model types are to be trained and compared, each model has its own set of parameters and hyperparameters. In this case we would need a test set to determine if the parameters are adequately trained, then a second validation set is needed to monitor performance improvements resulting from hyperparameter configuration adjustments. Note that a back and forth process is needed here, and that these two processes are not executed as sequential batch processes. If the same test set were used to test both the parameter and hyperparameter adjustments, then the model will become overly specialised at making predictions on the test set. Similarly, a third test set will be needed to compare the performance of the three model types. Whenever a model is adjusted in response to a data set, it is as though information from the data set leaks into the model, which necessitates the use of a reserved data set to ensure that the information leakage, so to speak, did not deteriorate the model's ability to generalise.

Although this approach of increasing the number of test sets is technically possible, it is often practically infeasible due to the fact that labelled data sets are limited for various problems which limits the amount of data that can be reserved for all the required test sets. Besides, even if there is enough data, it still does not solve the first of the two discussed problems associated with the 'hold-out set' validation method. K-fold cross-validation provides the means of addressing both these problems.

With K-fold cross-validation a data set is randomly divided into K subsets, known as folds. Consider the case where the folds are labelled 1 to K. Then, during hyperparameter optimisation, for each new configuration, the model is trained on folds $Y = K \setminus \{a\}$, and the resultant model is tested using fold a. This process is repeated for $a = 1, 2, \dots, K$ and the final model performance is calculated as the average performance attained over all K iterations [38].

K-fold cross-validation reduces the high variability risk associated with relying on a single test set by averaging model performance over K test sets. By doing so we get a more stable estimate of true model performance. Secondly, K-fold cross-validation allows for any number of

model aspects to be trained and tested using only a single train and test set, which results in a large reduction in data wastage in the form of reserved test sets. The hold-out set and K-fold cross-validation workflows are illustrated in Figure 18 and Figure 19 respectively.

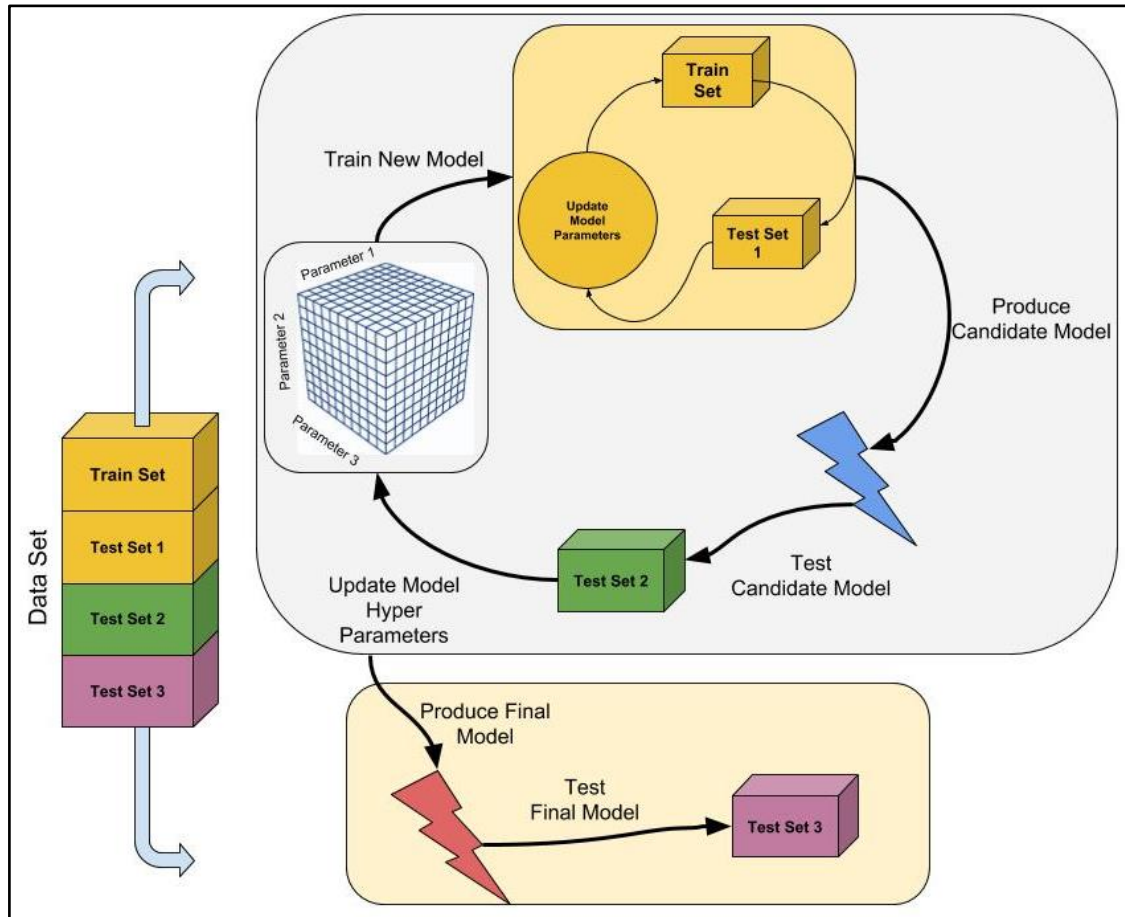


Figure 18: Illustration of hold-out set model validation process

Figure 18 illustrates the hold-out set modelling and validation workflow for a model type that includes three hyperparameters. The illustration depicts the available data set that has been sectioned as follows: A training set to be used by a training algorithm to tune the model parameters; a test set to measure model performance outside of the initial training set; a second test set to measure the performance of the model over competing configurations of its hyperparameters; and finally, a test set to test the model with the best performing hyperparameter configuration, as measured on the second test set.

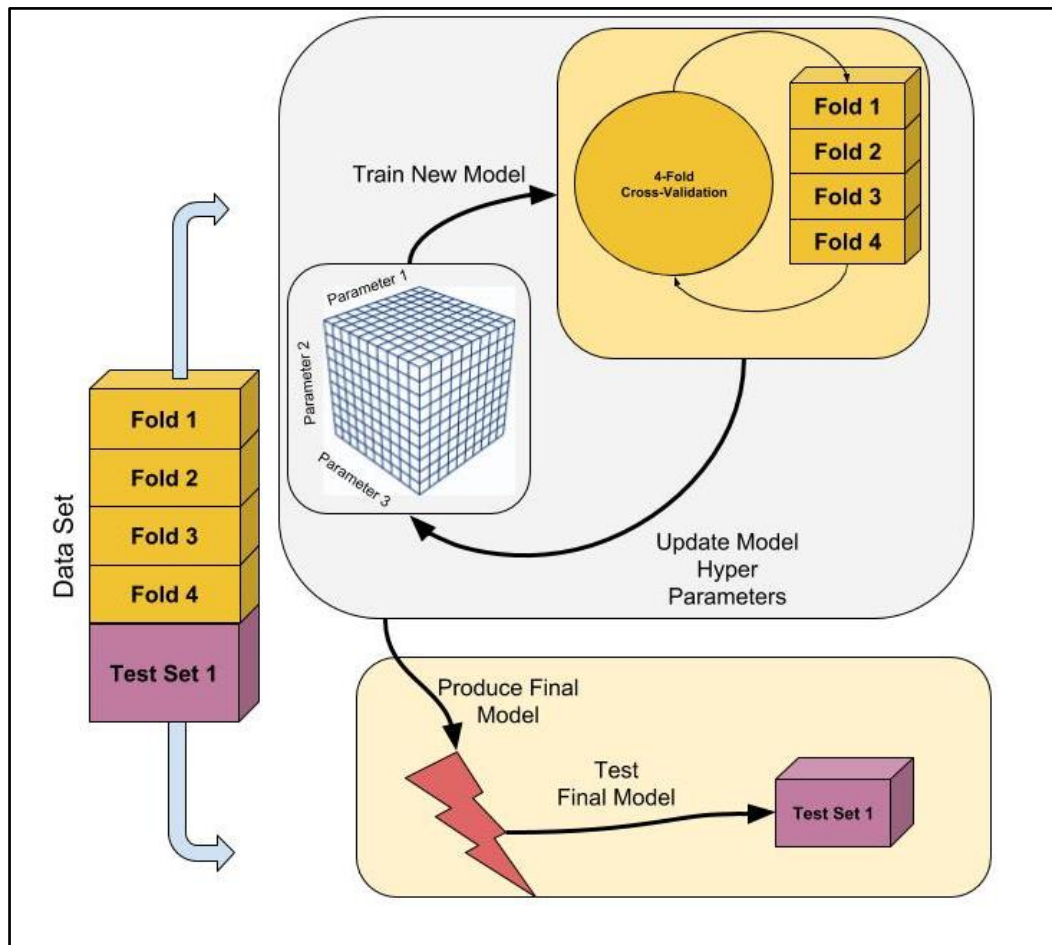


Figure 19: Illustration of K-fold cross-validation process

Figure 19 illustrates how the K-fold cross-validation method would be applied for the same modelling task. In reality this method divides a data set into two sections: a section required for model optimisation, which will be used to perform the K-fold cross-validation, and a test set to test the final model. The advantage of K-fold cross-validation lies in the fact that this will remain the case irrespective of the number of model aspects that require optimisation. Conversely, with hold-out set validation, the required number of test sets grows linearly with the number of aspects that require optimisation.

It should be noted that K-fold cross-validation is not without its drawbacks. One important disadvantage, for instance, is that K-fold cross-validation is computationally much more expensive than hold-out set validation. However, the severity of this disadvantage is determined by how urgently an accurate model must be produced, and will diminish over time as the cost of computing power decreases.

3.3 Chapter summary

Chapter 3 provides the findings from the literature that was reviewed for the completion of this research. The chapter begins with a review of approaches to physical asset maintenance. Pros and cons of each of the reviewed approaches were highlighted. It was found that a maintenance strategy based on data-driven prognostics allow for timely notification of asset failure, as well as, maintenance planning and resource optimisation. However, data-driven prognostics requires that relevant data must be available to build accurate prognostic models.

The chapter continues with a formal definition of ML and a review of various types of ML problems was provided. A framework for supervised ML problems was formulated. The three ML model types used in this research, namely logistic regression, ANNs and random forest, were reviewed. Each of the ML model types were described in terms of how they work, how they are developed and their advantages and disadvantages.

The chapter continues with a report on methods that are used to evaluate the performance of classification ML models. Finally, the chapter concludes with a description of the methods that are used to optimise ML model hyperparameters.

The following chapter provides detail on the methods that were used to build the ML models for the case study of this research.

4 ML model development

This chapter serves to document the process by which the ML techniques discussed in Section 3.2 were implemented on the Metrorail wheel condition monitoring data. Details regarding the table schema and field definitions of the condition monitoring data are provided in Table 2. A summary of Metrorail's wheel condition monitoring data will be provided. The output of the summary will inform how the data will be cleaned and formatted to create input features and target variables. Finally, the ML techniques discussed in Section 3.2 will be used to develop a logistic regression, ANN and random forest model using the cleaned and formatted wheel condition monitoring data.

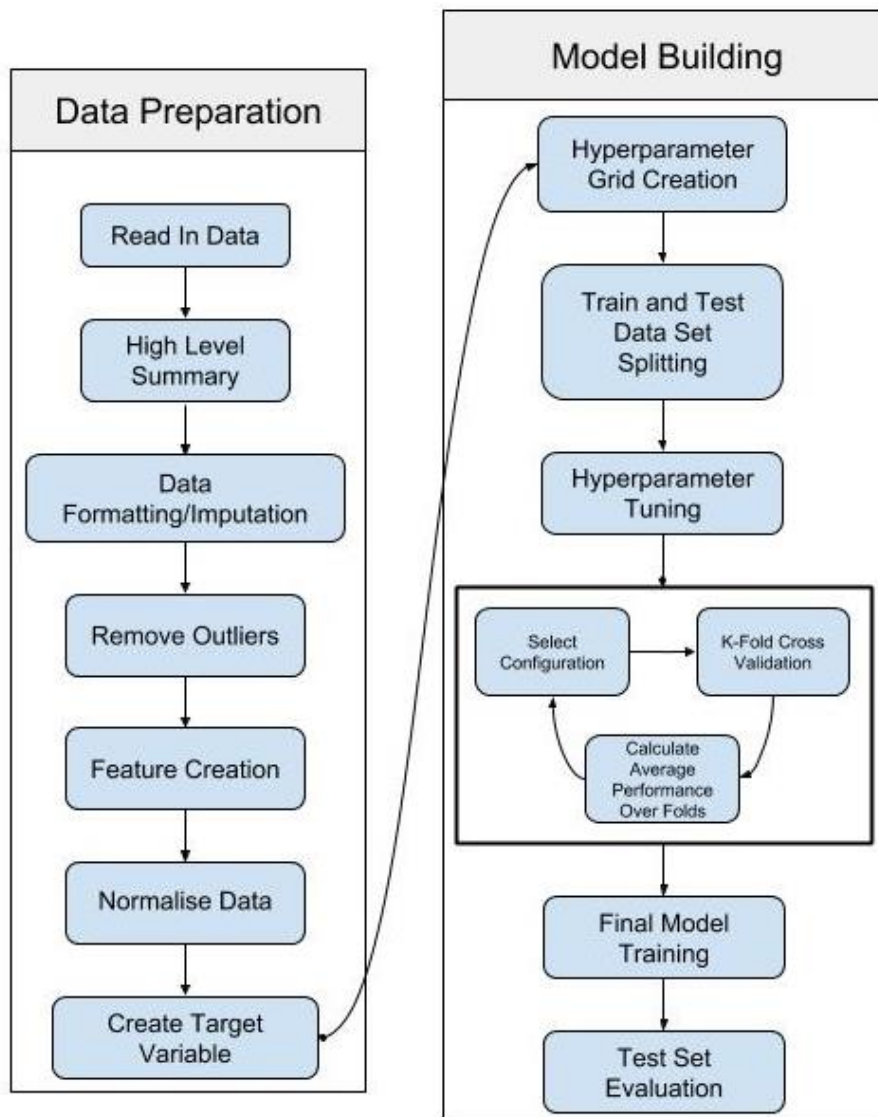


Figure 20: ML model development process diagram

The ML model development process followed in this chapter is depicted in Figure 20. The process was implemented using *R*, a free software environment for statistical computing. The implemented *R* code is provided in Appendix A.

4.1 Data preparation

This section presents the steps followed during the data preparation phase of the ML model development process, as shown in Figure 20.

4.1.1 Data read in and high level summary

The wheel condition monitoring data set (from here on referred to as ‘the data set’) was received as a comma separated value file from Metrorail’s rolling stock department, and read into *R*. The data set consisted of 454’187 records, following the schema provided in Table 2. A summary of the data set is provided in Table 5.

The summary revealed that various fields contained missing values. The missing values in the ID field were caused by erroneous measurements that were deleted and repeated. Initially, the measurement timestamp only included the date of the measurement, which resulted in the missing values in the Time field. Finally, the summary revealed that the Date and Time fields were both in *Microsoft Excel*’s serial date format, which is why these fields are both integer fields instead of being in the correct date and time formats.

Table 5: Initial summary of wheel data

| Field | Data Type | Range/Categories | NA Value Count |
|--------------|----------------------|---|----------------|
| Date | Integer | 40’156 – 43’286 | 0 |
| Time | Continuous Numerical | 0.0006 – 0.9998 | 1’369 |
| Stock | Categorical | 10M2, 10M2T, 10M3, 10M3T, 10M5, 10M5t, 10M5T, 5M2A, 5M2At, 5M2AT, 8M, 8MT | 0 |
| Set Number | Categorical | 10M2, 10M2T, 10M3, 10M3T, 10M5, 10M5t, 10M5T, 5M2A, 5M2At, 5M2AT, 8M, 8MT | 0 |
| Bogie Number | Categorical | 1, 2 | 0 |

Table 5: Initial summary of wheel data (Continued)

| Field | Data Type | Range/Categories | NA Value Count |
|-------------|----------------------|------------------------|----------------|
| Axle Number | Categorical | 1, 2, 3, 4 | 0 |
| Wheel ID | Categorical | 1, 2, 3, 4, 5, 6, 7, 8 | 0 |
| FT | Continuous Numerical | 0.00 – 38.34 | 69'510 |
| TD | Continuous Numerical | -5'224.6 – 9'851.0 | 2'579 |
| FS | Continuous Numerical | 0.00 – 30.52 | 181'744 |
| FH | Continuous Numerical | 0.00 – 50.27 | 3 |
| HW | Continuous Numerical | -14.59 – 0.00 | 69'510 |
| HW | Continuous Numerical | -14.59 – 0.00 | 69'510 |

4.1.2 Data formatting and imputation

The first fields that required reformatting were Date and Time. The Date field was converted to a *yyyy-mm-dd* format, while only the hour value of the time field was used to convert it to an *hh:00:00* time format. The missing values in the Time field had to be imputed in order for the Date and Time fields to be combined, so as to form a single Datetime (TD) field. The median hour value for the Time field was used to impute the missing values because, unlike the mean, it avoided the need to deal with fractions of hours. The median is also generally considered to be a robust statistical measurement, making it a good choice for integer imputation.

After combining the formatted Date and Time fields, R's built-in date function was used to convert the field to a timestamp. The summary of the newly created Datetime field revealed that the data set contained measurements that were recorded between 2009-12-09 09:00:00 and 2018-07-05 15:00:00, which is what was expected.

The final formatting step was to convert the categorical variables from long format to wide format. This entailed the creation of Boolean variables, which represent each category of the categorical variables, that indicate if a record belonged to a category or not. This is done so that categorical variables can be recoded into 1's and 0's, which can be used by the ML models. The effect of this transformation is illustrated in Appendix B.

4.1.3 Outlier correction

4.1.3.1 First phase outlier removal

Outliers in the data set were mostly introduced by faulty wheel measurements. For instance, the data set included negative TD measurements, which were clearly errors. Each wheel wear measurement has a specific range within which it must fall in order for the wheel to be operable. This range specifies the decommissioning thresholds for each of the wear measurements, as well as the maximum allowable values for new wheels. These values are listed in Table 3, and they were used as an initial guideline to clean up and remove outliers from the wheel wear measurements.

The first phase of outlier removal was initiated by creating a flag for each wheel wear measurement, that indicates whether a particular measurement was logically impossible. It was decided to flag these measurements before removing them in order to ascertain whether some of the data could be salvaged. For instance, a record might contain a faulty TD measurement while also containing a correctly measured FH measurement. The rules used to flag measurements were as follows: a) all measurements must be greater than zero (except for HW, which must be less than or equal to zero; b) measurements may not be empty or contain an *NA* value. After the flags were created, a check was done to find all records that were positively flagged as erroneous for all of its wheel wear measurements. These records were then dropped from the data set.

4.1.3.2 Wheel instance identification

One of the major challenges presented by the data set was that it did not contain a field that uniquely identified wheel instances. The term 'wheel instance' is used here to refer to a unique physical wheel that was employed by Metrorail, as opposed to 'wheel ID', which is a term that forms part of the nomenclature used to specify a unique railway wheel position on a train set. Therefore, it is a wheel instance that is installed, used, worn out or damaged and then replaced by another wheel instance. It is essential to mark unique wheel instances in order to model wheel wear and to create wheel prognostic models.

The process of creating a wheel instance identification field is included in the outlier removal section of this project, because the two processes are related, and the main challenge presented by both these processes is the same. For both outlier removal and wheel instance identification the main challenge is to separate noise in the data from the true signal. The reason why wheel instance identification is done before fixing or removing outliers is because a wheel replacement will cause a spike in wheel wear measurements that might be mistaken for an outlier or erroneous measurement.

The following figures are used to illustrate this statement. Figure 21 is a graph of raw FH measurements over time for a wheel position, that was sampled from the data set. The graph contains three clear anomalies which are encircled in red. Anomalies 1 and 3 are clearly measurement errors that sporadically broke the wheel wear trend. However, anomaly 2 was caused by a wheel replacement, which lead to a resetting of the wheel wear level, after which the wheel wear trend continued normally.

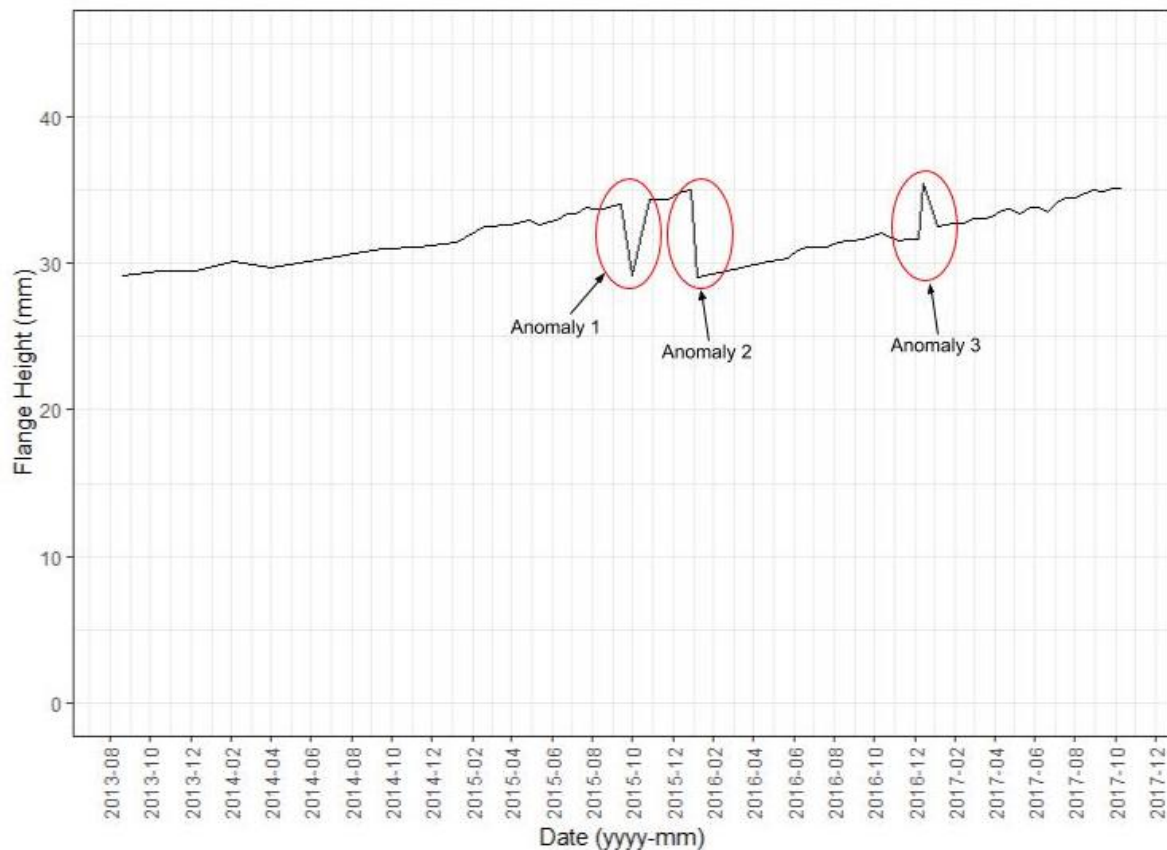


Figure 21: Raw FH sample measurements with anomalies indicated

If an automated outlier imputation algorithm were to be implemented on this data sample before wheel instance identification, it would possibly clean up the data to take the form shown in Figure 22. Such a data transformation will certainly introduce erroneous wheel wear data, seeing as the FH measurement goes far beyond the allowed 35 mm mark. This will negatively impact the accuracy of any ML models that are developed using this data. Figure 23 illustrates what Figure 22 would have looked like if the individual wheel instances were flagged before the outliers were fixed.

By flagging unique wheel instances, the outlier imputation process is able to ignore the apparent wear measurement anomalies caused by wheel replacements. This makes it easier for outlier

identification and imputation procedures to be focused on actual outliers caused by faulty measurements, which improves the quality of the ML model development data set and, therefore, the accuracy of the developed ML models.

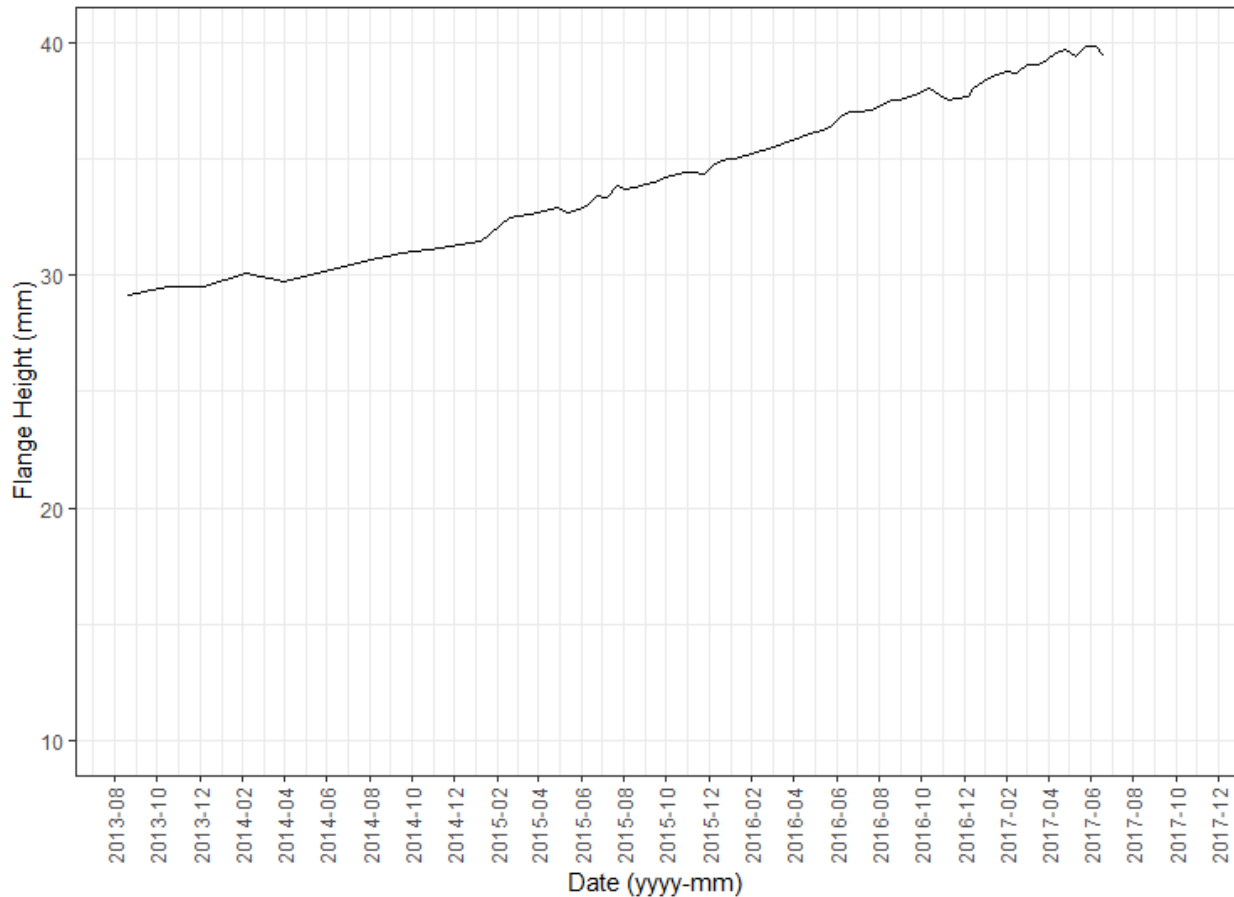


Figure 22: Cleaned FH sample measurements without wheel instance identification

FH was selected as the measurement used to aid in identifying wheel replacements. The flagging process involved analysing certain aspects of changes in wheel FH measurements over time as well as characterising the typical change observed in FH measurements by a wheel replacement. A sample of ten wheel positions was drawn from the data set to conduct this analysis. Each sampled wheel position had to include at least 20 measurements to increase the possibility of observing the normal wear trend as well as a wheel replacement in each sample. The analysis was initialised by drawing the ten samples, containing a total of 339 observations, and cleaning the FH measurements of each sample by hand. The first step in the clean-up process was to impute all obvious outliers, such as those annotated as Anomaly 1 and 3 in Figure 21, by using the average of the two measurements adjacent to the outlier.

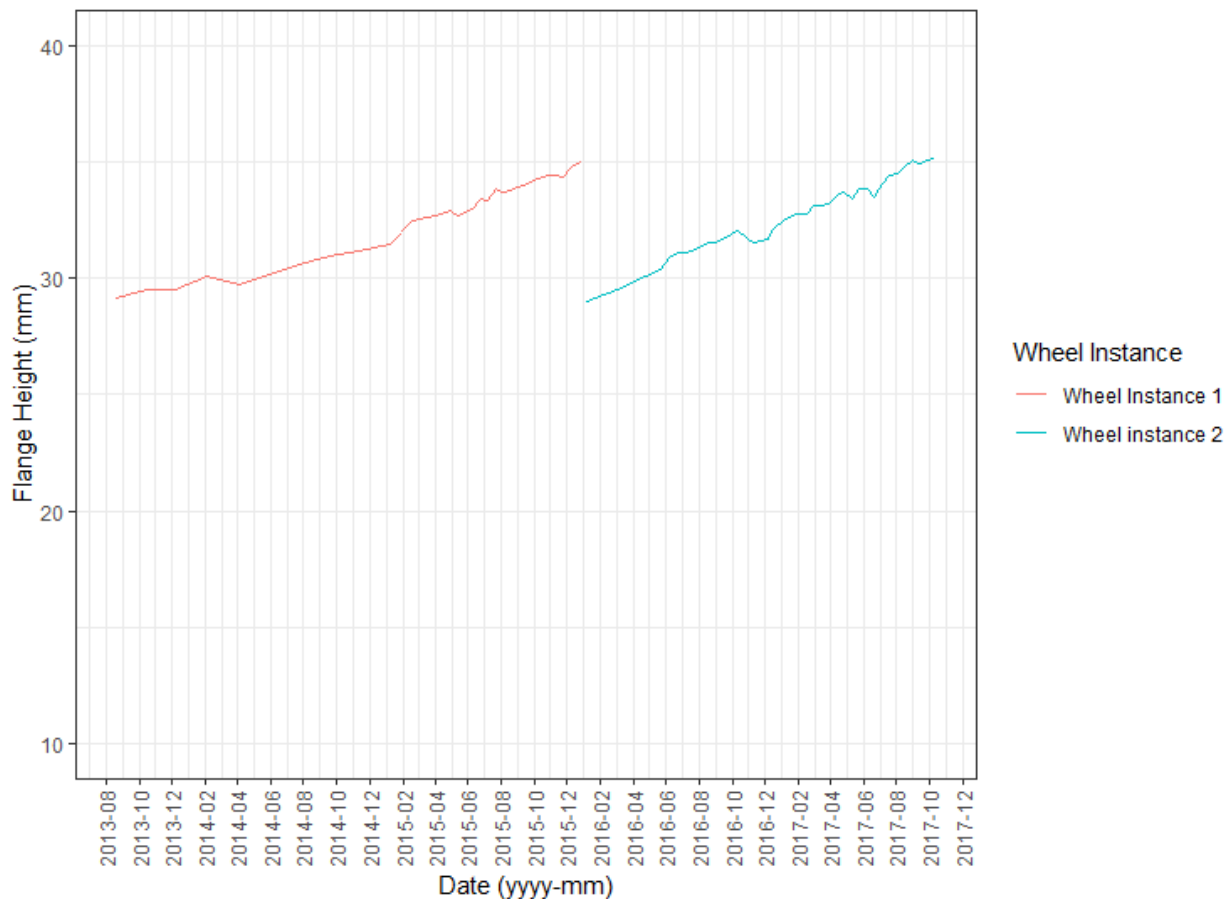


Figure 23: Cleaned FH sample measurements after wheel instance identification

The second step in the wheel instance identification process was to split the sample data set into wheel instances by identifying anomalies in the data similar to Anomaly 2 in Figure 21, where there is a drop in FH, after which the normal wheel wear trend continues, indicating a wheel replacement. Each sample was then split into wheel instances at these anomalies, as was done to create the data illustrated in Figure 23.

The third step was to analyse the FH difference between consecutive measurements. This entailed creating a variable, FH difference, comprised of the differences between consecutive FH measurements in the data samples. The FH difference had to be normalised because the time between wheel measurements was not constant, which created outliers in the FH difference in cases where there was a large time difference between consecutive measurements. FH difference was normalised by creating a time difference variable comprised of the difference in time between consecutive wheel measurements, measured in weeks. FH difference was then divided by the time difference variable to get a normalised average weekly FH difference variable. A box plot was produced from the average weekly FH differences in order to visualise this variable's distribution and outliers. The box plot is shown in Figure 25.

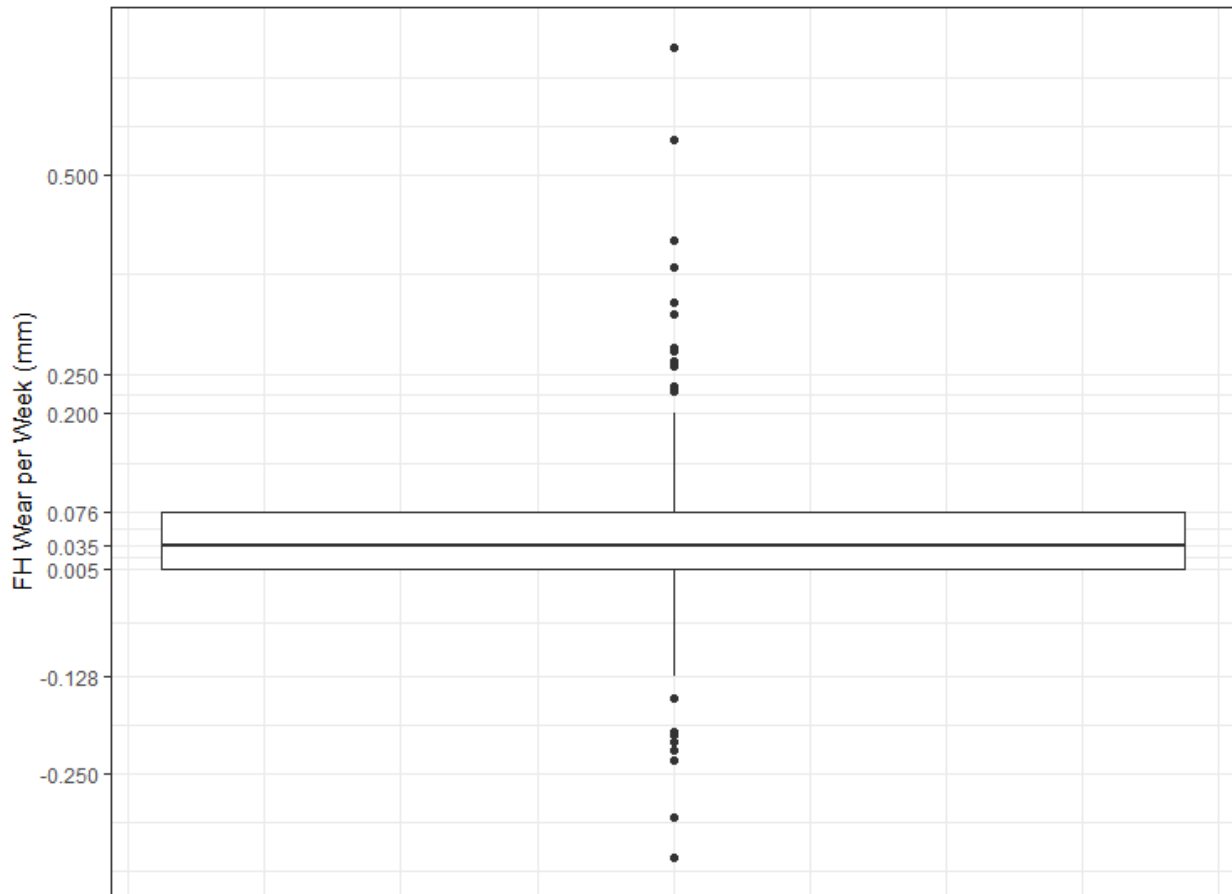


Figure 24: Box plot of average weekly FH wear of data sample

Figure 25 shows that the interquartile range of the data samples' average weekly FH wear is 0.071mm and that the first and third quartiles are at 0.005mm and 0.076mm respectively. The information provided by the normalised weekly FH wear measurements was useful for creating rules to identify positive FH outliers; however, negative FH outlier identification called for a separate investigation. The reason for this is that one can expect a positive relationship between time and FH, which is why it remains sensible to make use of insights gained from FH wear measurements that are normalised with respect to time, but only if the FH values are increasing over time. Decreasing FH measurements, on the other hand, are only expected when wheel replacements occur. It is therefore nonsensical to make use of these normalised FH wear measurements to identify negatively valued FH wear outliers, because any time that passes between the last measurement of a replaced wheel and the first measurement of the replacement wheel will result in shrinking the expected drop in FH, potentially to the point where it may be treated as a faulty measurement instead of a wheel replacement.

To address this issue, an analysis was done on all cases where there was a decrease in FH for subsequent measurements. The goal of the analysis was to find a cut-off value for determining

when a drop in FH is large enough to signify a potential wheel replacement. This was done by producing a box plot of all the negative observations of consecutive measurement FH differences. The box plot is illustrated in Figure 25. The extreme point of the lower whisker of the box plot, at the -0.417 mm mark, was selected as the cut-off point for identifying instances where wheel changes occurred.

A final peculiarity found in the FH data was the occurrence of subsequent FH measurements that represent a slight decrease in FH, with no clear spike in the data to indicate that the measurement is faulty. This phenomenon is shown in Figure 21, where one can see that there are instances, not including the annotated anomalies, where the FH appears to decrease between consecutive measurements. This is due to the fact that the wheel is not measured at exactly the same point on its running surface each time a measurement is taken, therefore local imperfections on the wheel's running surface cause this apparent wear reversal. In order to clean up such occurrences, a conservative pocketing method was devised. In the FH case, this method dictated that, if a measurement is larger than its subsequent measurement, then the subsequent measurement is changed to the former of the two measurements. This process is conservative in the sense that it will always default to the measurement that indicates a greater extent of deterioration.

A combination of the aforementioned pocketing method and the information provided by the box plot in Figure 25 was used to create and execute a procedure to clean up the FH measurements in the original wheel condition monitoring data set. The procedure consisted of the following steps: 1) Split the data set to create unique Stock, Set Number, Coach Number, Bogie Number, Axle Number and Wheel Number combination sets; 2) Remove all sets containing fewer than 10 FH measurements; 3) Sort each set by Datetime, from oldest to newest; 4) Loop through each set and, at each data point, calculate the FH difference divided by the time difference (measured in weeks) between consecutive measurements, to create a relative change value. If the relative change is positive and it lies between the extreme points of the box plot whiskers, no adjustment or wheel change flagging takes place. If the relative change is positive and it lies outside of the box plot whisker, then the relative change for the subsequent data point is reviewed. If the revision shows that the subsequent relative change is negative and it lies outside of the box plot whisker, then an occurrence similar to anomaly 3 in Figure 21 took place. The former measurement is then deemed to be an outlier, and its value is imputed using the average of the point or points adjacent to the outlier, and no wheel change flagging takes place. Alternatively, the wheel is assumed to have sustained excessive damage or wear and no adjustments or wheel change flagging takes place. If the relative change is negative, then the FH difference value is inspected. If the FH difference is greater than -0.417 mm, then the FH measurement is

set to that of the former data point. If the FH difference is less than or equal to -0.417 mm, then the relative change for the subsequent data point is reviewed. If the revision shows that the subsequent relative change is positive and it lies outside of the box plot whisker, then an occurrence similar to anomaly 1 in Figure 21 took place. The former measurement is deemed to be an outlier, and its value is imputed using the average of the point or points adjacent to the outlier and no wheel change flagging takes place. Otherwise, an occurrence similar to anomaly 2 in Figure 21 took place. The data point is flagged as a point where a wheel change took place and no FH value adjustments are made.

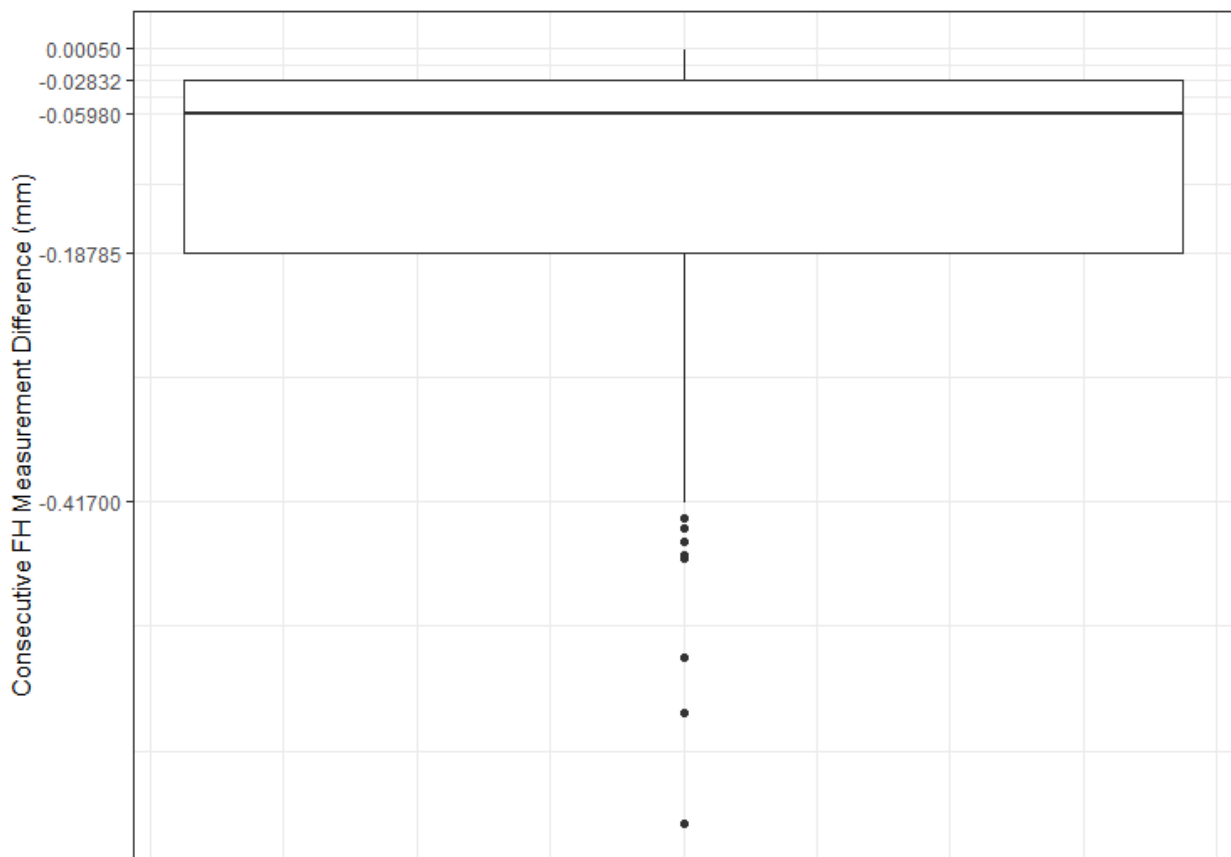


Figure 25: Box plot of negative consecutive FH measurement differences of data sample

Three wheel-position samples were drawn from the data set to check whether the wheel instance identification process worked as expected. The effect of the wheel identification process on these samples is illustrated in Figure 26 to Figure 31. These samples were arranged from least to most challenging in terms of the FH outlier removal and wheel instance identification. Two graphs were produced per sample, the first of which was to illustrate whether the procedure was successful in removing outliers and faulty measurements, and the second was to illustrate whether the procedure was successful in identifying wheel replacements and labelling wheel instances. Figure 26 shows that the procedure corrected an outlier and two faulty

measurements, while Figure 27 shows that the procedure correctly detected no wheel replacements and only labelled one wheel instance. Figure 28 shows that the procedure corrected an outlier and a faulty measurement, while Figure 29 shows that the procedure correctly detected one wheel replacement and labelled two wheel instances. Finally, Figure 30 shows that the procedure corrected multiple outliers and faulty measurements, while Figure 31 shows that the procedure correctly detected two wheel replacement and labelled three wheel instances. The results of these samples indicated that the outlier detection and wheel instance identification procedure worked as desired.

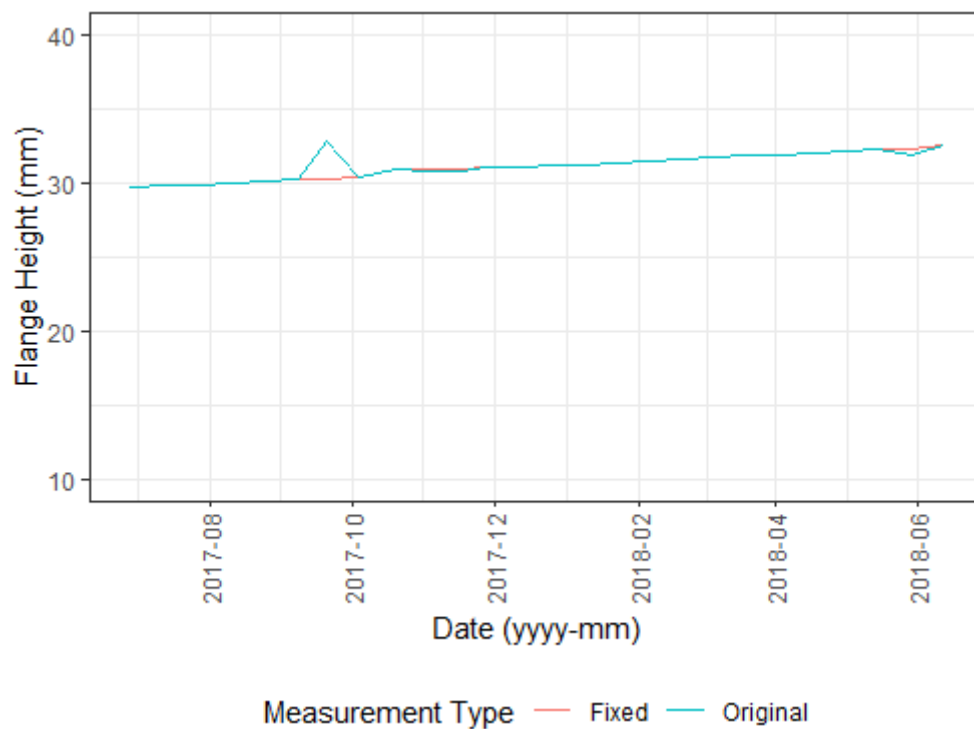


Figure 26: Original FH vs. fixed FH for data sample 1

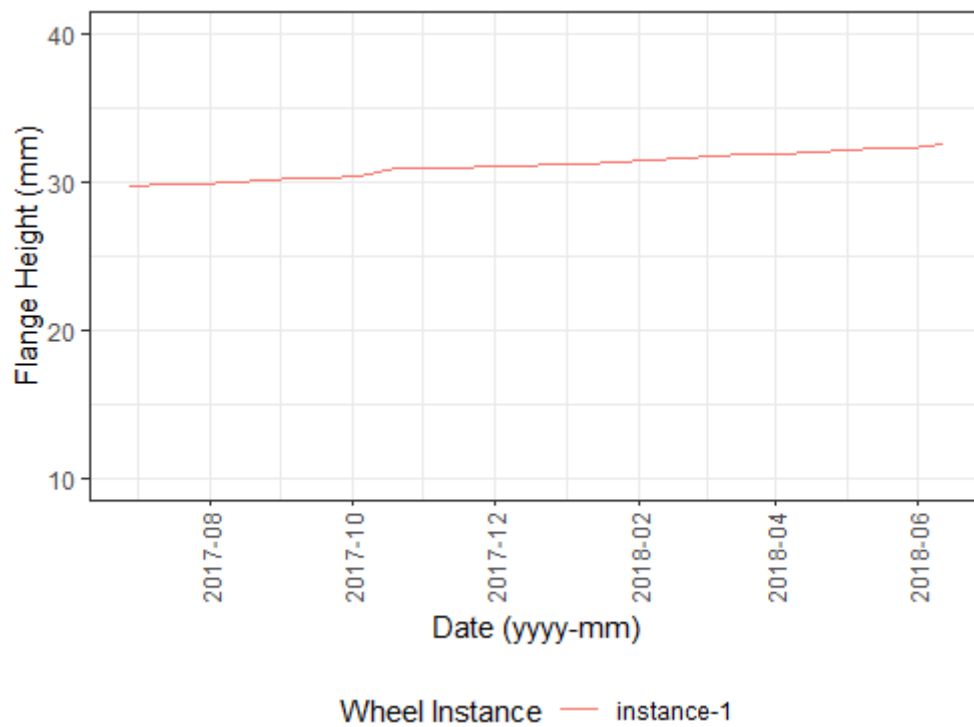


Figure 27: Fixed FH for data sample 1 grouped by wheel instance

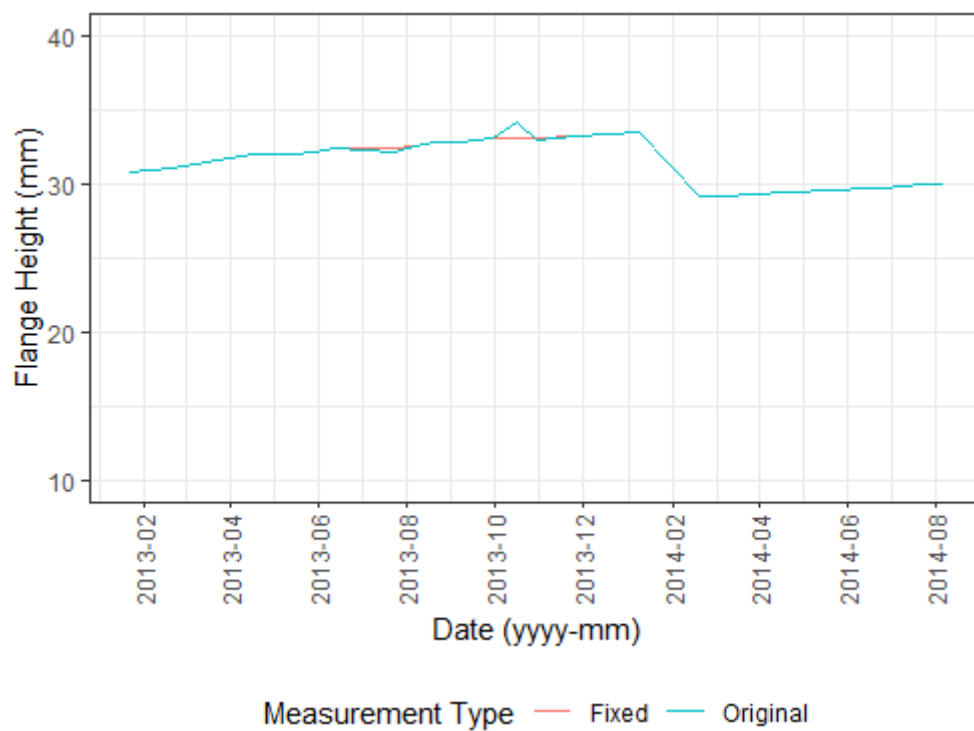


Figure 28: Original FH vs. fixed FH for data sample 2

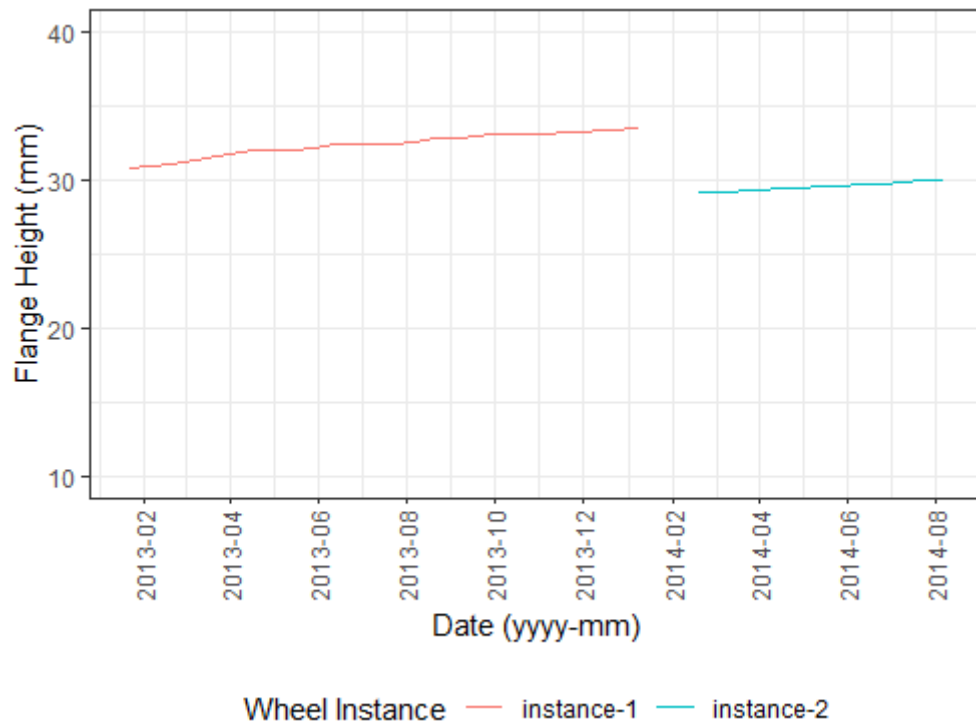


Figure 29: Fixed FH for data sample 2 grouped by wheel instance

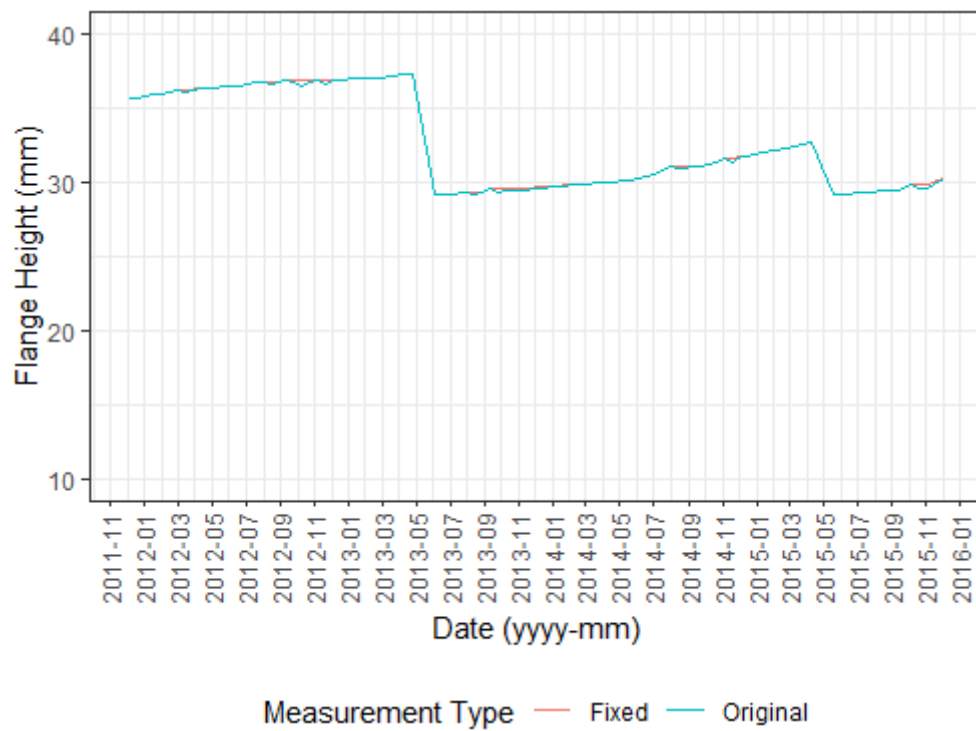


Figure 30: Original FH vs. fixed FH for data sample 3

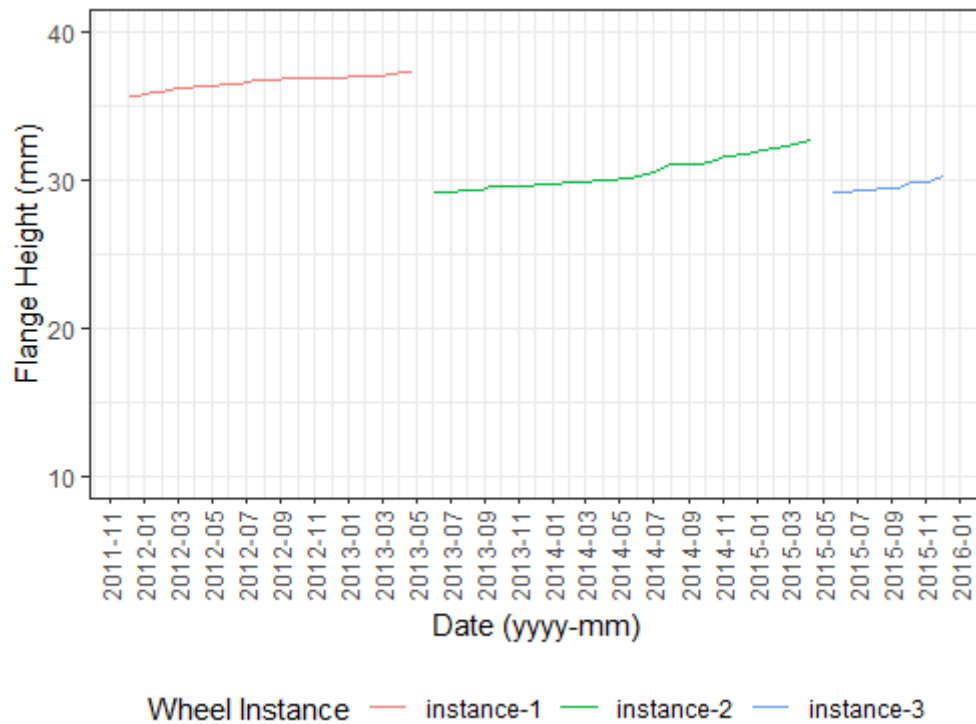


Figure 31: Fixed FH for data sample 3 grouped by wheel instance

4.1.4 Feature engineering

Feature engineering is the process of applying domain-specific knowledge in order to manipulate and combine features in a ML data set to create new features that are potentially more informative. The intention is to create features that will provide additional information to ML algorithms which will aid in creating more accurate ML models [49]. The selection of additional features that were created for this specific project was based on domain-specific knowledge that was obtained during site visits at the Metrorail rolling stock maintenance workshop. Firstly, it was noted that the demand for Metrorail's services fluctuated during the year, which had an influence on the rate of rolling stock usage, and therefore, the railway wheels. Secondly, it was advised that the elapsed time between rolling stock inspections should be used as a proxy for the distance covered by a given set of rolling stock, seeing as the data pertaining to the actual distance covered was not readily available for analytical purposes [12]. Thirdly, it was noted that the relative diameter of wheels on a given coach was an area of concern for Metrorail's maintenance management. This is due to the fact that differences in the diameter of wheels on a railway coach had an impact on how the weight of the coach was distributed among the wheels. Finally, it was thought that the extent of wear a wheel exhibited during an inspection provided information as to whether the wheel will exceed the allowed extent of wear at the point in time when the following inspection takes place. These domain-specific insights were applied to Metrorail's wheel wear data set in order to create additional input features. In the following

subsections the engineered features created in this project, as well as the reasoning behind each of these features will be discussed.

4.1.4.1 Time between measurements

Time between measurements was defined as the amount of time that has passed between wheel condition monitoring interventions and it was measured in days. This feature was chosen to serve as a proxy for the extent to which a wheel was used between consecutive condition monitoring interventions. The reasoning behind the creation of this feature was the fact that the extent to which a wheel has deteriorated since its previous condition assessment is certainly related to the distance covered by the wheel in the elapsed time. Due to the fact that the actual distance covered by a wheel instance between consecutive wheel inspections was not available in the provisioned data set, the elapsed time between consecutive wheel inspections was used as an approximation for the distance covered.

4.1.4.2 Wheel instance age

Wheel instance age was defined as the age of a wheel instance, measured in days. This feature was created to provide the ML algorithms with information related to possible non-linearity in the extent of wheel wear over time. The reasoning behind the creation of this feature was that if one were to hold all other variables fixed, then the difference in the change in age of a wheel might change the wheel's wear rate due to various effects such as annealing, tempering, cold working and oxidation. Effects such as these change the hardness of the wheel's running surface and can accumulate over time, therefore changing the wear rate of a wheel over time.

4.1.4.3 Relative FH difference

Relative FH difference was used to create three features related to the difference in FH of neighbouring wheels. The reasoning behind the creation of these features was that an imbalance in the diameter of neighbouring wheels on a train might lead to an imbalance in how the weight of a train is distributed among the wheels, where wheels with larger diameters will carry more weight than others. FH was used as a measure of wheel diameter, instead of TD, because FH was an actual measurement whereas TD was a calculated feature that was extrapolated by the *MiniProf*, making it a less reliable measurement than FH. The three created relative FH difference features were bogie FH difference, axle set FH difference, and average bogie FH difference.

Coach set FH difference was calculated for a given wheel as the difference between the wheel's FH and that of the smallest FH measurement on the same coach. Axle set FH difference was calculated for a given wheel as the FH difference between the wheel and the smallest FH

measurement on the same axle. Average coach FH difference was calculated as the difference between the average FH for a coach and that of the neighbouring coach with the smallest average FH.

4.1.4.4 Moving average wear rate

Moving average wear rate was calculated as the average wear rate for a given wheel wear measurement over the past three measurements, measured in mm/day. This feature was created to provide the ML algorithms with information regarding the rate at which a given wheel is deteriorating according to the various wear measurements. This feature was also created to make up for the fact that the data set does not contain any information related to the condition of the rails on which the wheels run. Deteriorated rails will certainly have an impact on the rate at which the wheels deteriorate, therefore, moving average wheel rate was created to make up for this missing information.

4.1.4.5 Month of year

The month of year feature was defined as the months of the year that have passed between consecutive wheel inspections for a given wheel instance. The motivation for this feature stemmed from the notion that fluctuations in demand for rail transport during the year might result in increases and decreases in scheduled trips and passengers per trip. These fluctuations in usage could possibly impact the wear rate for a wheel for a given time period, which is valuable information that the ML algorithms can use to build prognostic models.

4.1.4.6 Previous measure value

This engineered feature was created for each of the wheel wear measurements. The reasoning behind this feature was that the level of a given measurement at the time of the previous condition monitoring intervention will provide useful information as to whether the given measurement will have exceeded its tolerated threshold, as per Table 3, at the time of the subsequent condition assessment. The information provided by the previous measure values was expected to be especially insightful in conjunction with the information provided by the time between measurements feature and the moving average wear rate. This is because the previous measurement value informs on the point of departure from the previous condition assessment, the wear rate informs on the rate at which the wheel deteriorates per day and the time between measurements informs on the time that has passed since the previous wheel condition assessment.

4.1.5 Feature normalisation

Feature normalisation is the process of rescaling numeric features so that they all have the same scale. This data transformation is a prerequisite for most ML algorithms that make use of a distance measure during model learning or value predictions. Examples of such ML algorithms include algorithms that make use of gradient descent, such as logistic regression and ANN models, and clustering ML models, such as k-Nearest Neighbours, that assigns data as a member of the nearest class based on Euclidean distance.

Data sets often include features that are on vastly different scales. The data set used in this project, for instance, is comprised of features measured in days, millimeters and millimeters per day. The reason why it is important to ensure that these features are on the same scale is because most ML algorithms do not interpret the units of the input features, but only the magnitude of the values. Therefore, variations in large-scaled features could have a disproportionately large influence on the produced ML models.

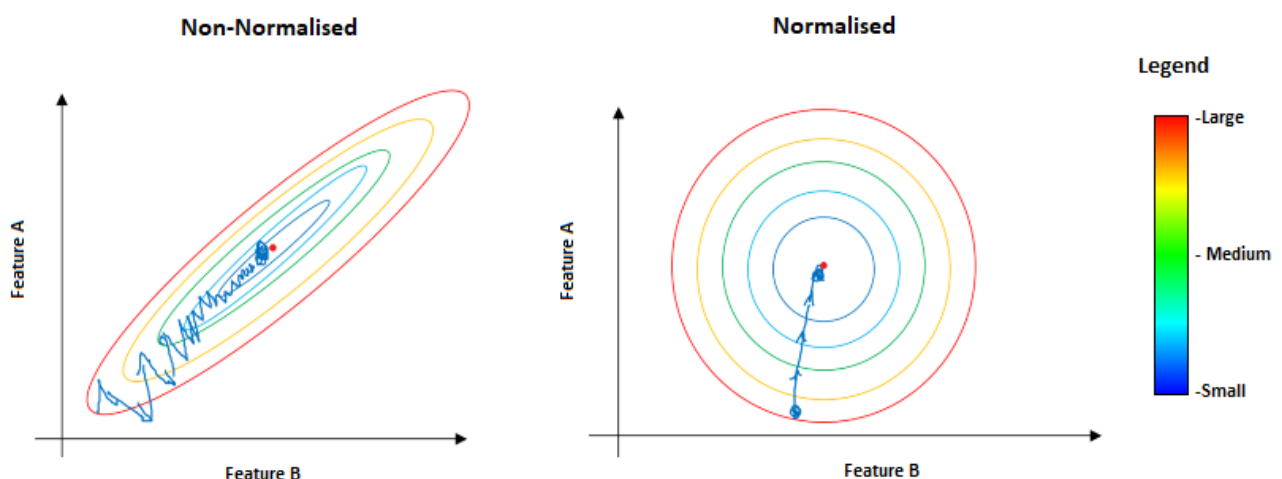


Figure 32: Example of ML model error contour plots for non-normalised and normalised input features [50]

Ensuring that input features are on the same scale is also good practice when gradient descent is used for model training. The reason for this can be explained with the help of Figure 32 [50]. The figure depicts two contour plots of the error surface for an ML problem with respect to two input features, A and B. In the first case, the features are left unnormalised, resulting in an elongated contour plot, while in the second the features are normalised, resulting in a symmetric contour plot. The markings on both charts show the path a gradient descent algorithm would follow to find the point where the error is smallest. One can see that the elongated error plot in

the unnormalised case forces the gradient descent algorithm to take a much longer path to the minimum point than in the normalised case. This means that gradient descent may take much longer to find an optimum set of model parameters if the features are not scaled beforehand.

$$z_i = \frac{x_i - \mu}{\sigma} \quad (14)$$

Z-normalisation was used to normalise the numeric features in the data set. This form of normalisation rescales the numeric variables so that they have the characteristics of a standard normal distribution, with a mean of 0 and a standard deviation of 1. Eq. 14 was used to implement the Z-normalisation. In Eq. 14, z_i is the i^{th} normalised feature value and x_i is the i^{th} feature value for i ranging from 1 to the number of records in the data set. Finally, μ is the mean of the feature and σ is the standard deviation of the feature.

4.1.6 Target variable creation

As stated in Section 3.2, the target variable, in the context of ML, is the variable comprised of observations generated by the target function, which the ML algorithm seeks to emulate through learning from examples. The output of this attempt at emulation is the ML model. It is important that the target variable be defined in such a way that the resultant ML model will be able to address the problem that the modeller aims to solve. The target variable dictates, to a large extent, what the ML model will be able to predict. It is useless to produce a ML model that can make accurate predictions if the predictions do not contribute to solving the problem that initiated the modelling process to begin with, hence the importance of correctly defining the target variable.

The values provided in Table 3, which contains the decommissioning thresholds for each of the wheel wear measurements, was used to define the target variables for this project. The logic used to create the target variables was as follows: First, the crossing direction in Table 3 was checked. This specification indicates whether a given wear measurement increases or decreases as the wheel deteriorates. The second step was to check the cut-off values for each of the wear measurements, which indicates the point at which a wear measurement has reached the point where it signifies that the wheel ought to be decommissioned. Then this information was used to formulate the criteria by which a binary target variable flag was created for each wear measurement. For cases where the crossing direction was 'decreasing', the criteria stated that if the wear measurement was less than or equal to the cut-off value, the binary variable was set to 1, otherwise it was set to 0. For cases where the crossing direction

was ‘increasing’, the criteria stated that if the wear measurement was greater than or equal to the cut-off value, the binary variable was set to 1, otherwise it was set to 0.

One could ask why the binary target for each wear measurement was not simply set to 0, save for the last observation for each wheel instance, seeing as this point indicates where a wheel replacement took place. The reason why this was not the case was due to the fact that the goal of this project was not model the replacement behaviour of Metrorail’s maintenance teams, but rather to model the fluctuations of the various wear measurements relative to their respective cut-off values. This is done because, although Metrorail has clearly defined cut-off values for each wear measurement, the Maintenance teams do not always adhere to these specifications due to supply chain constraints etc. This means that some wheels are allowed to run well past the point where it should have been replaced, according to the various wear measurements. Defining the target variables in the aforementioned way provides a standardised framework for the modelling process, as opposed to modelling the occurrence of actual wheel replacements in the data, which could have occurred in a certain manner due to a wide range of very case specific reasons.

4.2 ML model building

This section describes the process followed to build the logistic regression, ANN and random forest ML models. A high level depiction of the process is illustrated in the *Model Building* diagram in Figure 20. This section will describe each of the phases depicted in the aforementioned diagram, for each of the three model types.

4.2.1 Train and test data set splitting

This section describes the *Train and Test Data Set Splitting* phase of the ML model building procedure. The reason why this phase was not described in the context of each of the model types was because the steps involved were identical for each of the model types.

R’s built in *sample* function was used to randomly divide the data set into a train and test data set. 70% of the data set was allocated to the train data set and the remaining 30% was allocated to the test data set. The *sample* function accepts a data set and the proportion of the data set which should be randomly sampled from the set, which, in this case, was set as the wheel measurement data set and 0.7 times the size of the data set, respectively. The remainder of the data set was assigned to be the test data set. The attractive property of the *sample* function is that it attempts to maintain the proportions of categorical variables as it samples. This reduces

the risk accidentally assigning all observations for a specific class of a categorical variable to either the test or the train data sets.

4.2.2 Logistic regression model

Of the three model types that was used during this project, logistic regression was the simplest to train and test because there were no hyperparameters that needed to be adjusted explicitly. The *glm* function in R was used to create the logistic regression models for each of the wheel wear measurements, with the *glm* function call configured as follows:

```
logistic_model = glm(train$target_variable~.,family = binomial(link = "logit"),data = train)
```

As shown in the function call, the link function for the logistic model was specified to be the logit function.

For each wheel wear measurement, the model was trained on the train data set and evaluated on the test data set. The evaluation was done based on the performance metrics discussed in Section 3.2.5.2. In the following subsections, the results for these evaluations will be provided for each of the wheel wear measurements.

4.2.2.1 FH model evaluation

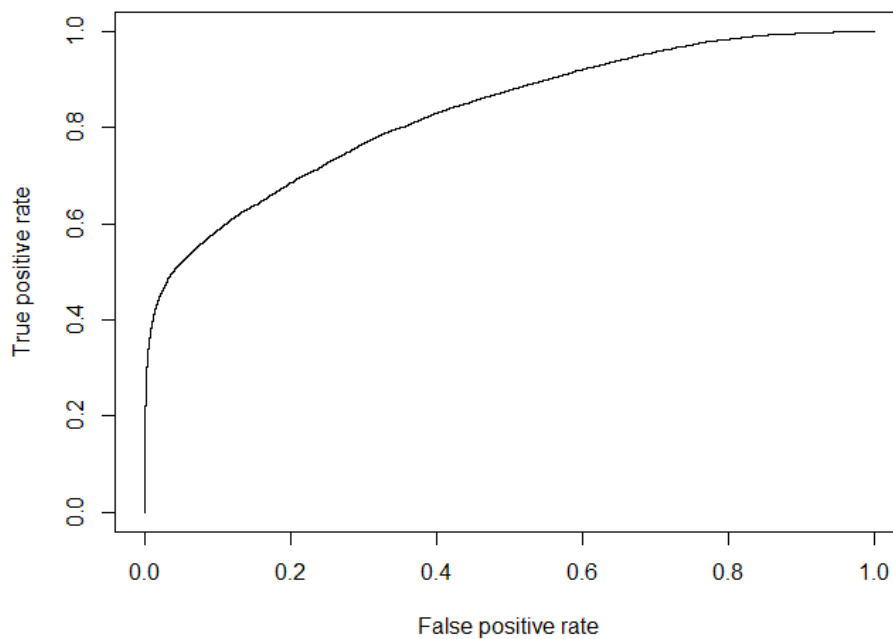
The confusion matrix for the FH wear prognostic model is provided in Table 6. From Table 6, the various accuracy metrics discussed in Section 3.2.5.2 were calculated and are provided in Table 7. An ROC curve, which is illustrated in Figure 33, was produced for the model performance on the test set using the R function *ROCR*. The associated AUC value was 0.831.

Table 6: Confusion matrix for logistic regression model of FH wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 3'633 | 6'405 |
| Actual Class = 0 | 514 | 73'350 |

Table 7 :Confusion matrix metrics for logistic regression model of FH wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.362 |
| Specificity | 0.993 |
| Accuracy | 0.917 |
| F1 | 0.954 |

**Figure 33: ROC curve for logistic regression model of FH wear prognostics**

4.2.2.2 TD Model evaluation

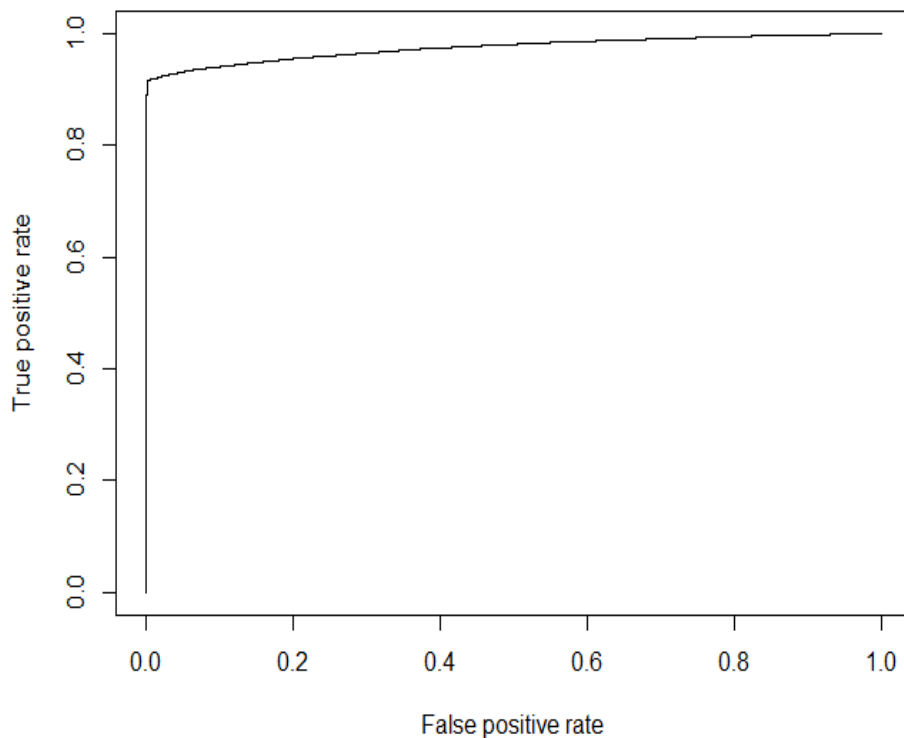
Similarly to FH, the confusion matrix for the TD wear prognostic model is provided in Table 8 and the metrics discussed in Section 3.2.5.2 are provided in Table 9. An ROC curve, which is illustrated in Figure 34, was produced for the model performance on the test set using the R function *ROCR*. The associated AUC value was 0.975.

Table 8: Confusion matrix for logistic regression model of TD wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 37'944 | 3'454 |
| Actual Class = 0 | 91 | 41'413 |

Table 9: Confusion matrix metrics for logistic regression model of TD wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.917 |
| Specificity | 0.998 |
| Accuracy | 0.957 |
| F1 | 0.959 |

**Figure 34: ROC curve for logistic regression model of TD wear prognostics**

4.2.2.3 HW model evaluation

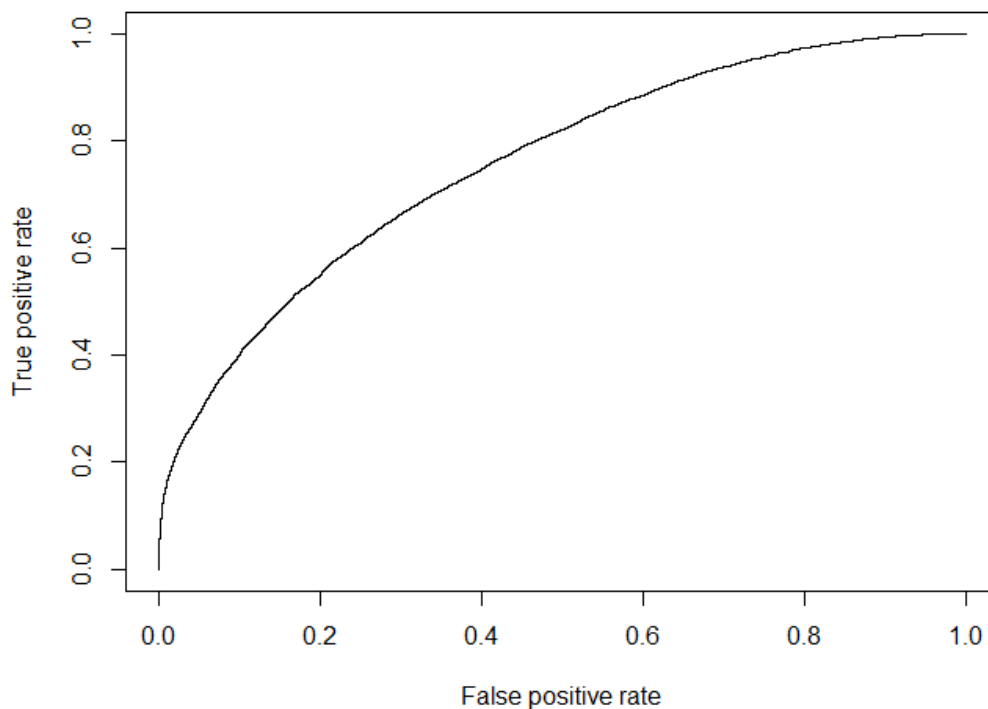
The confusion matrix for the HW wear prognostic model is provided in Table 10 and the metrics discussed in Section 3.2.5.2 are provided in Table 11. An ROC curve, which is illustrated in Figure 35, was produced for the model performance on the test set. The associated AUC value was 0.757.

Table 10: Confusion matrix for logistic regression model of HW wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 619 | 6'481 |
| Actual Class = 0 | 218 | 75'584 |

Table 11: Confusion matrix metrics for logistic regression model of HW wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.087 |
| Specificity | 0.997 |
| Accuracy | 0.919 |
| F1 | 0.958 |

**Figure 35: ROC curve for logistic regression model of HW wear prognostics**

4.2.2.4 FS model evaluation

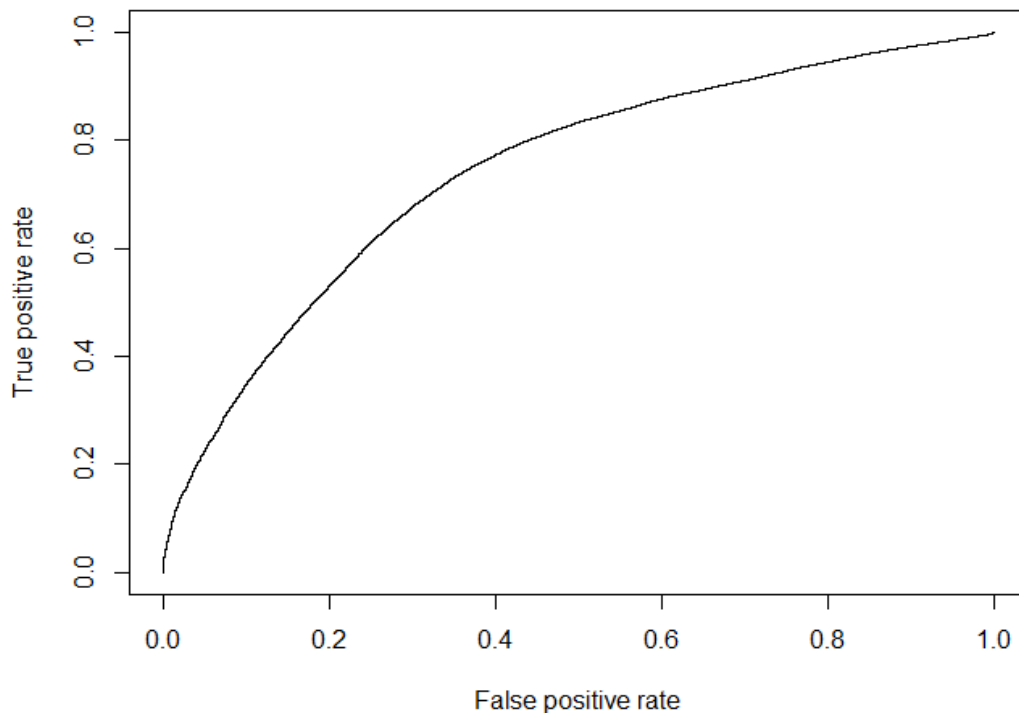
The confusion matrix for the FS wear prognostic model is provided in Table 12 and the metrics discussed in Section 3.2.5.2 are provided in Table 13. An ROC curve, which is illustrated in Figure 36, was produced for the model performance on the test set. The associated AUC value was 0.742.

Table 12: Confusion matrix for logistic regression model of FS wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 10'821 | 16'409 |
| Actual Class = 0 | 6'805 | 48'867 |

Table 13: Confusion matrix metrics for logistic regression model of FS wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.397 |
| Specificity | 0.878 |
| Accuracy | 0.720 |
| F1 | 0.482 |

**Figure 36: ROC curve for logistic regression model of FS wear prognostics**

4.2.2.5 FT model evaluation

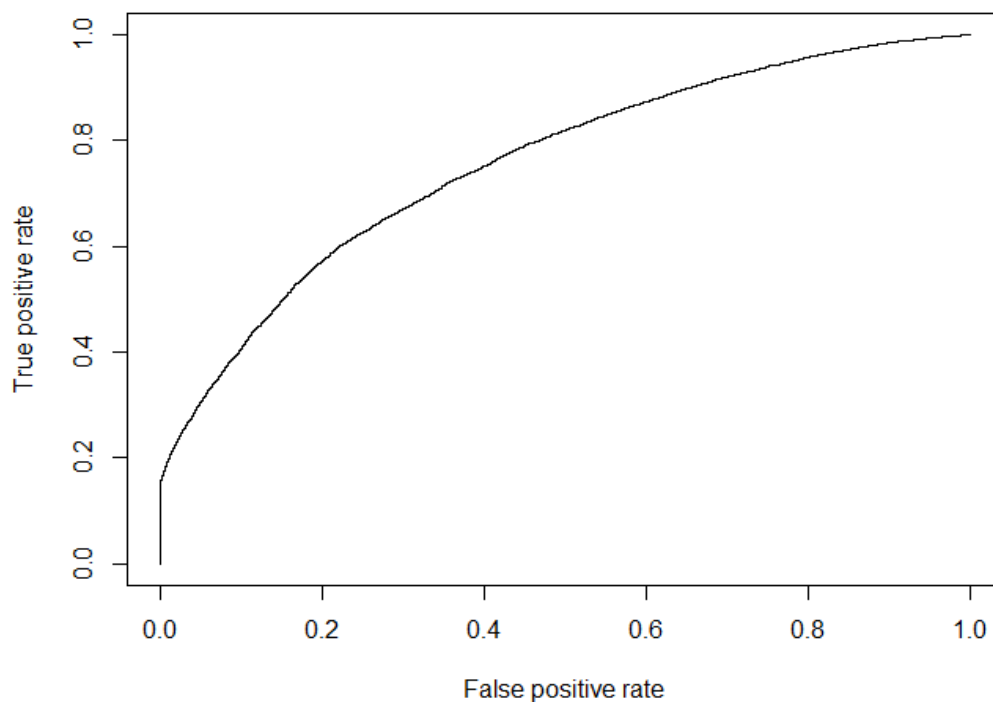
The confusion matrix for the FT wear prognostic model is provided in Table 14 and the metrics discussed in Section 3.2.5.2 are provided in Table 15. An ROC curve, which is illustrated in Figure 37, was produced for the model performance on the test set. The associated AUC value was 0.756

Table 14: Confusion matrix for logistic regression model of FT wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 1'150 | 6'256 |
| Actual Class = 0 | 73 | 75'423 |

Table 15: Confusion matrix metrics for logistic regression model of FT wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.155 |
| Specificity | 0.999 |
| Accuracy | 0.924 |
| F1 | 0.985 |

**Figure 37: ROC curve for logistic regression model of FT wear prognostics**

4.2.2.6 Logistic regression summary

The logistic regression model performed well when it came to providing FH prognostics. The model achieved an accuracy of over 90% and an AUC of greater than 0.8. The ROC curve was indicative of a healthy model. It had a smooth curve which tended toward the point (0,1), before bending away toward the point (1,1). This indicates that the model was capable of separating the target variable classes with high accuracy.

The model performed exceptionally well in terms of TD prognostics. It achieved an accuracy of over 95% and an AUC of over 0.95. The ROC curve was indicative of an extremely performant model that is capable of accurately separating the target variable classes. The ROC curve tended sharply toward the (0,1) point before cornering toward the (1,1) point. This jagged shape of the curve raises concerns over the behaviour of the model and could indicate that the output target variable classes were either very easily separable, or the data exhibited strange behaviour.

The model struggled to perform well when it came to HW prognostics. Although it achieved an accuracy of over 90%, it had an exceptionally low sensitivity rate of less than 10%. This indicates that the model struggled to predict the positive cases of the target variable. The model's incapability to separate the target variable classes is reflected in its ROC curve, which is relatively straight and flat. Furthermore, the model achieved a relatively low AUC of 0.753. The poor performance might be due to the relatively biased nature of the logistic regression model.

The model also struggled to perform well in terms of FS prognostics. The model achieved a relatively low accuracy rate of 72% as well as a relatively low AUC of 0.742. The ROC curve was also relatively straight and flat, indicating that the model had trouble separating the target variable classes. Again, the poor performance might be due to the relatively biased nature of logistic regression.

Finally, the logistic regression model performed moderately well on the FT measurements. The model achieved an accuracy rate of 92%, however, its AUC was only 0.756. This indicates that the high accuracy was strongly attributed to target variable class imbalance, and that the model still struggled to identify cases with a positive target variable class.

4.2.3 ANN model

The ANN models were built with the help of the *ANN2 R* package. This package makes the *neuralnet* function available, which accepts various hyperparameters as well as the set of input features and the target variable as parameters. The following section describes the process by which the five ANN models were developed, as per the Model Building phase depicted in Figure 20.

4.2.3.1 Hyperparameter grid creation

The hyperparameters that were optimised during the development of the ANN models were the number of hidden layers in the network, the number of neurons per hidden layer, the momentum coefficient, η , and the learning rate, α . The initiation of the network weights was dealt with by the

ANN2 package, which initiates the weights to random values, the range of which is dictated by the number of input features.

Table 16: ANN hyperparameter grid

| Hyper-parameter | Option 1 | Option 2 | Option 3 | Option 4 | Option 5 | Option 6 | Option 7 | Option 8 |
|-------------------|----------|----------|----------|----------|----------|----------|----------|----------|
| η | 0.1 | 0.2 | 0.3 | 0.5 | N/A | N/A | N/A | N/A |
| α | 0.005 | 0.050 | 0.500 | N/A | N/A | N/A | N/A | N/A |
| No. Hidden Layers | 1 | 2 | N/A | N/A | N/A | N/A | N/A | N/A |
| Nodes Layer 1 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| Nodes Layer 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |

The selected hyperparameters were optimised using a grid search process. An array of possible values was created for each of the hyperparameters. These values are provided in Table 28. The default value of η , as per *R*'s *ANN2* package is 0.2. A range of values around this default value was used to check if there is a value for η that performs better than the default value. The parameter α was allowed to range from 0.005 to 0.5, incremented by a factor of ten. The main objective in setting this parameter is to make it small enough to ensure that the prediction error will decrease with every iteration of gradient descent, while being large enough to allow gradient descent to converge in a reasonable time. The selected range and step size of α was deemed to be sufficient for finding a value for α that will satisfy this criterion. The number of hidden layers was limited to two, to allow the ANN to model complex functions, while also reducing the risk of overfitting. The number of neurons in the hidden layers was allowed to range between three and ten to provide the grid search algorithm with room within which it can search for a good performing ANN network configuration while also allowing the grid search process to execute within a reasonable amount of time.

4.2.3.2 Hyperparameter tuning

A combination of grid search and K-fold cross-validation was used to tune the ANN hyperparameters. The grid search was achieved, in *R*, by creating four sequentially nested for-loops, each of which looped through the array of one of the hyperparameters. No loop was created for the *No. hidden layers* parameter, instead, a value of zero was appended to the *Nodes Layer 2* parameter array, which forced the algorithm to omit the second hidden layer when this value was considered by the for-loop.

The K-fold cross-validation was achieved by creating a final for-loop within the grid-search for-loop structure. The procedure within this loop worked as follows: First, the training data set was shuffled and split into three equally sized parts. Then, the hyperparameters selected by the outer for-loops were passed as parameters to the *ANN2 neuralnet* function call, along with two-thirds of the shuffled training data set. The function then used the hyperparameter configuration and the two-thirds training data to build an ANN. The resultant ANN was then validated against the remaining third of the data set. This process was executed three times per hyperparameter configuration, which made it a threefold cross validation process. Finally, the average threefold cross validation F1 score was calculated to measure the performance of every hyperparameter configuration.

Each of the hyperparameter configurations was assigned with a unique process iteration identifier to keep track of how each configuration performed. Figure 38 illustrates the F1 score for each hyperparameter configuration that was tested during the FH ANN model development. The figure shows how the performance of the ANN models becomes highly unstable when the value for α is set to its highest value, which occurs exactly three times during the grid search process.

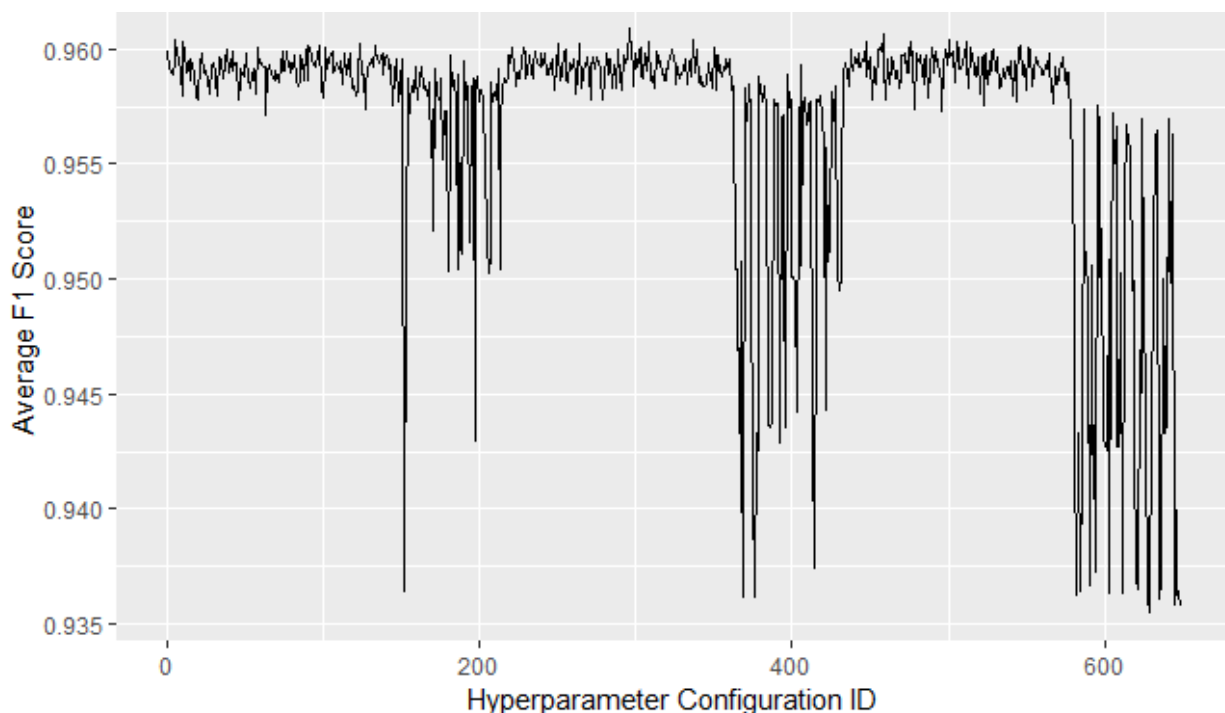


Figure 38: FH ANN hyperparameter grid search output

The best performing hyperparameter configuration was selected for each wear type. The final ANN configurations for each of the wheel wear measurements are provided in Table 17.

Table 17: Best performing ANN hyperparameters

| Measure | η | α | No. Hidden Layers | Nodes Layer 1 | Nodes Layer 2 |
|---------|--------|----------|-------------------|---------------|---------------|
| FH | 0.1 | 0.5 | 2 | 8 | 5 |
| TD | 0.3 | 0.05 | 1 | 3 | 0 |
| HW | 0.1 | 0.005 | 2 | 6 | 3 |
| FT | 0.3 | 0.05 | 2 | 7 | 6 |
| FS | 0.5 | 0.5 | 2 | 5 | 5 |

4.2.3.3 Final model training and evaluation

The final phase of the ANN model development process entailed training the ANN models for each of the wheel wear measurements on the entire train data set, with the hyperparameters set to those in Table 17. As with the logistic regression models, each model's performance was evaluated based on confusion matrix statistics, ROC curves and the AUC measure.

i) FH model evaluation

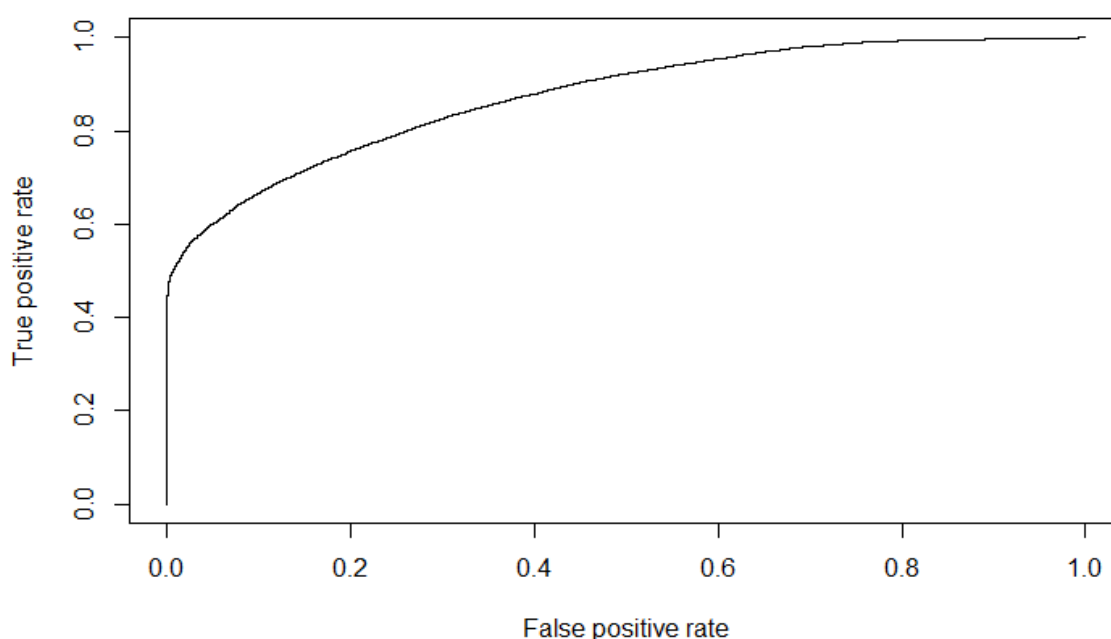
The confusion matrix for the FH wear prognostic ANN model is provided in Table 18. From Table 18, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 19. An ROC curve, which is illustrated in Figure 39, was produced for the ANN model performance on the test set. The associated AUC value was 0.874.

Table 18: Confusion matrix for ANN model of FH wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 4'908 | 5'130 |
| Actual Class = 0 | 319 | 72'545 |

Table 19: Confusion matrix metrics for ANN model of FH wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.489 |
| Specificity | 0.996 |
| Accuracy | 0.934 |
| F1 | 0.964 |

**Figure 39: ROC curve for ANN model of FH wear prognostics**

ii) TD model evaluation

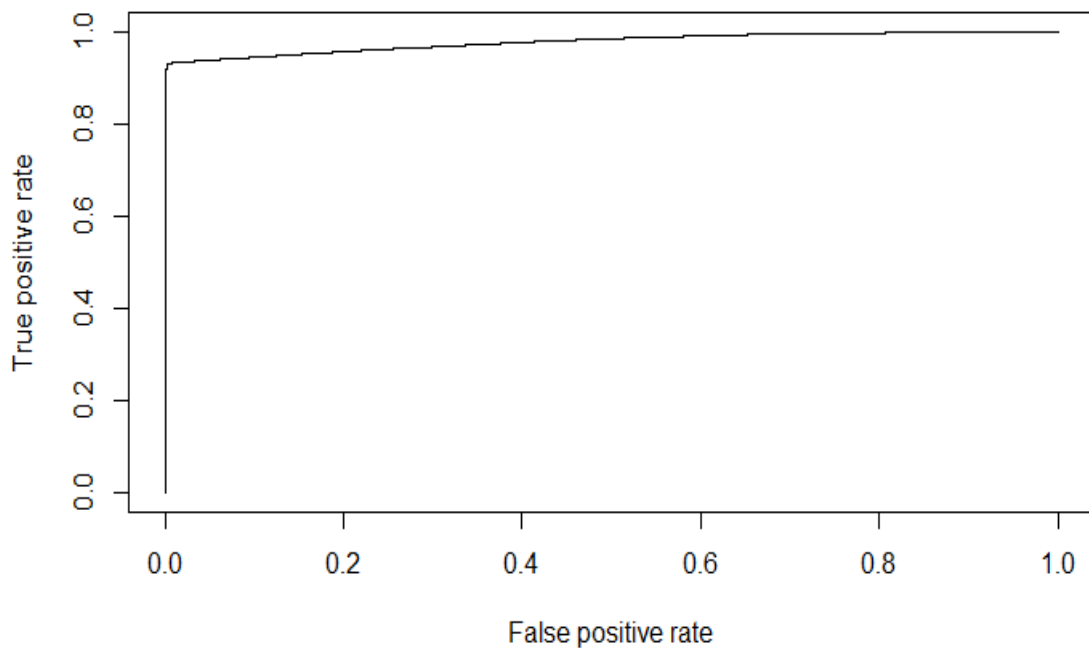
The confusion matrix for the TD wear prognostic ANN model is provided in Table 20. From Table 20, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 21. An ROC curve, which is illustrated in Figure 40, was produced for the ANN model performance on the test set. The associated AUC value was 0.977.

Table 20: Confusion matrix for ANN model of TD wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 38'500 | 2'898 |
| Actual Class = 0 | 148 | 41'356 |

Table 21: Confusion matrix metrics for ANN model of TD wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.930 |
| Specificity | 0.996 |
| Accuracy | 0.963 |
| F1 | 0.964 |

**Figure 40: ROC curve for ANN model of TD wear prognostics**

iii) HW model evaluation

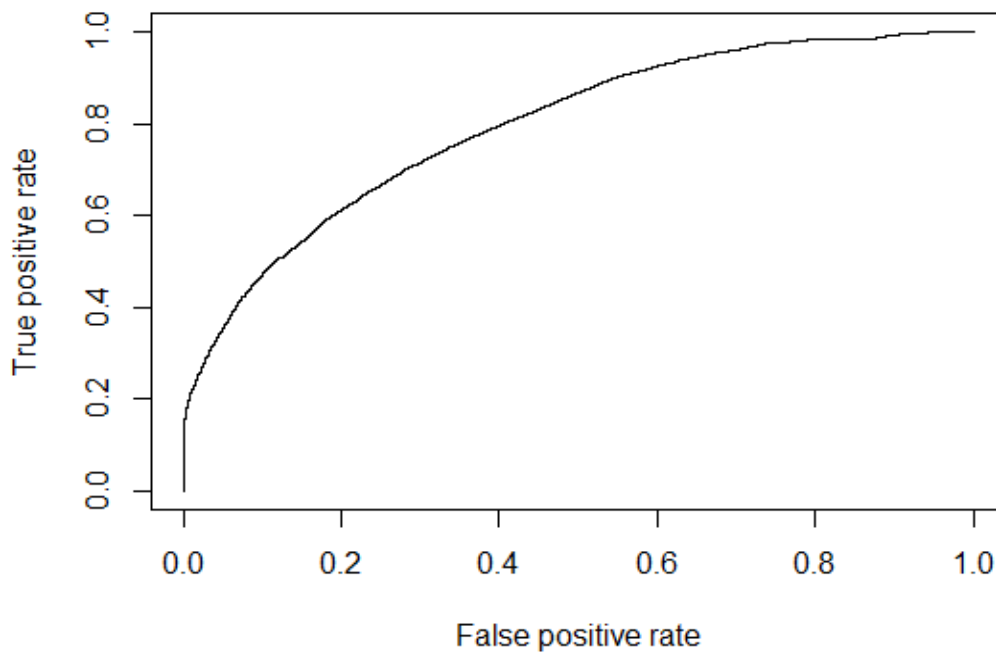
The confusion matrix for the HW wear prognostic ANN model is provided in Table 22. From Table 22, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 23. An ROC curve, which is illustrated in Figure 41, was produced for the ANN model performance on the test set. The associated AUC value was 0.794.

Table 22: Confusion matrix for ANN model of HW wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 1'390 | 5'710 |
| Actual Class = 0 | 510 | 75'292 |

Table 23: Confusion matrix metrics for ANN model of HW wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.196 |
| Specificity | 0.993 |
| Accuracy | 0.925 |
| F1 | 0.960 |

**Figure 41: ROC curve for ANN model of HW wear prognostics**

iv) FS model evaluation

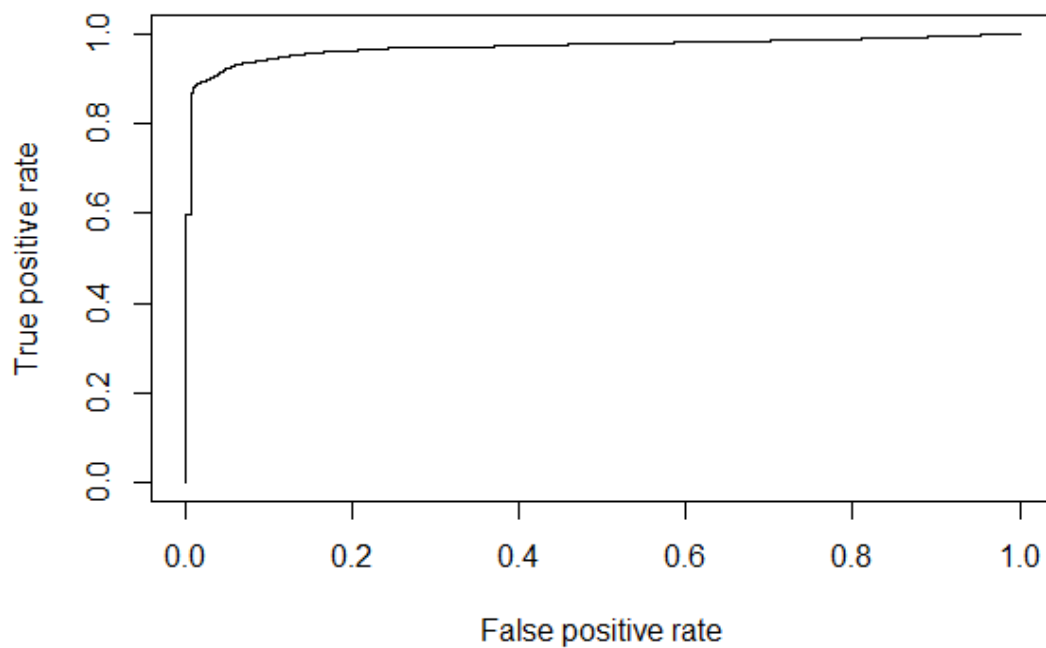
The confusion matrix for the FS wear prognostic ANN model is provided in Table 24. From Table 24, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 25. An ROC curve, which is illustrated in Figure 42, was produced for the ANN model performance on the test set. The associated AUC value was 0.969.

Table 24: Confusion matrix for ANN model of FS wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 50'171 | 5'501 |
| Actual Class = 0 | 870 | 26'360 |

Table 25: Confusion matrix metrics for ANN model of FS wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.901 |
| Specificity | 0.968 |
| Accuracy | 0.923 |
| F1 | 0.892 |

**Figure 42: ROC curve for ANN model of FS wear prognostics**

v) FT model evaluation

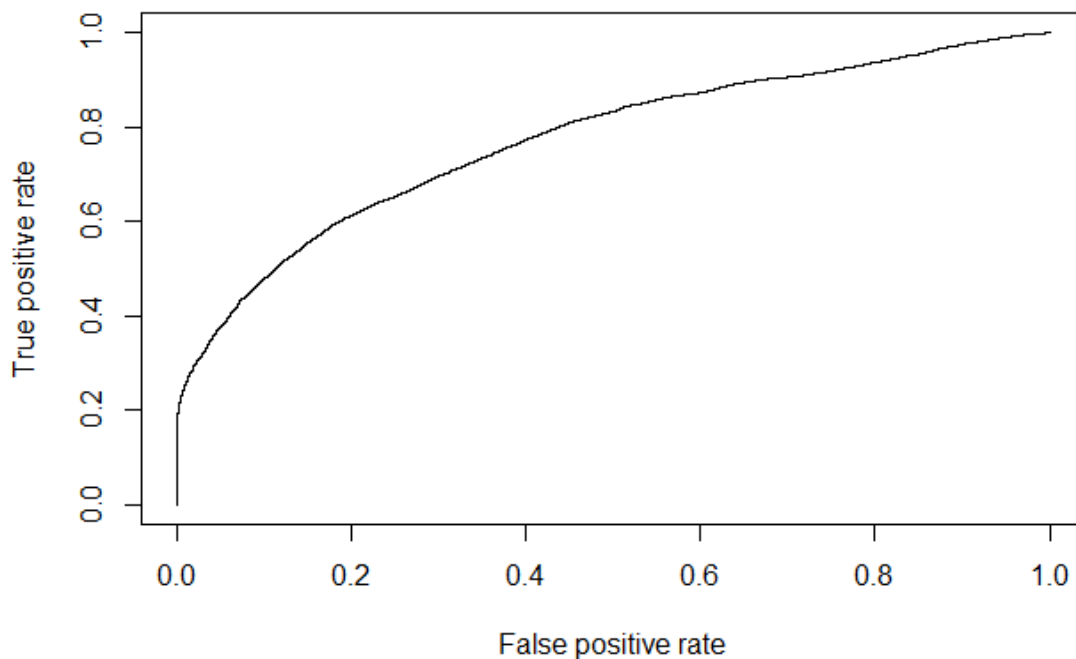
The confusion matrix for the FT wear prognostic ANN model is provided in Table 26. From Table 26, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 27. An ROC curve, which is illustrated in Figure 43, was produced for the ANN model performance on the test set. The associated AUC value was 0.772.

Table 26: Confusion matrix for ANN model of FT wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 1'677 | 5'729 |
| Actual Class = 0 | 380 | 75'116 |

Table 27: Confusion matrix metrics for ANN model of FT wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.226 |
| Specificity | 0.995 |
| Accuracy | 0.926 |
| F1 | 0.960 |

**Figure 43: ROC curve for ANN model of FT wear prognostics**

4.2.3.4 ANN performance summary

The ANN model type performed very well on the FH wear measurements. The model achieved an accuracy rate of over 90% and its AUC score was 0.87, which is high. The model's ROC curve was indicative of a healthy ML model, that is behaving as expected, with a smooth curve that tends toward the (0,1) point before curving toward the (1,1) point. This indicates that the model was able to separate the target variable classes well.

As with the logistic regression model, the ANN performed extremely well on the TD wear measurements. The model attained an accuracy rate of over 95% and had an extremely high AUC of 0.977. The extremely good performance leads to the same concerns that was raised with the logistic regression model, which was that the extremely good performance might be due to either easy separability of the TD target variable classes, or that the TD related data exhibited strange behaviour.

The ANN had difficulty with providing HW prognostics. Although the model achieved an accuracy rate of 92.5%, it had a relatively low sensitivity rate and, therefore, a relatively low AUC of 0.794. This indicates that the model performed moderately well when it came to separating the target variable classes.

The ANN model performed extremely well when it came to providing FS wear prognostics. The ROC curve tended to very near to the (0,1) point, before elbowing sharply toward the (1,1) point. This indicates that the model was capable of separating the target variable classes very well. The model achieved an accuracy rate of 92.3% and had an AUC of 0.969, which is high. Although the model performance and its ROC curve is similar to that of the TD case, the concern that was raised for the TD case is not raised here, because FS is a directly measured metric, as opposed to TD, which is an extrapolated metric. Therefore, the risk of a model achieving exceptional performance due to anomalies in the data is less in the FS case than in the TD case.

Finally, the ANN model performed moderately well when it came to FT wear prognostics. The model attained a high accuracy rate of 92.6%, however, the specificity rate was relatively low, at 22.6%. This indicates that the model struggled to separate the target variable classes. This notion is supported by the relatively flat shape of the ROC curve and its relatively low AUC of 0.727.

4.2.4 Random forest model

The random forest models were built with the help of the *ranger* R package. This package makes the *randomForest* function available, which accepts various hyperparameters as well as the set of input features and the target variable as parameters. The following sections describe the process by which the five random forest models were developed, as per the Model Building phase depicted in Figure 20.

4.2.4.1 Hyperparameter grid creation

The hyperparameters that were optimised during the development of the random forest models were the number of trees to grow (B), the number of variables to sample per tree (K), the size of the samples drawn per tree (L) as well as the minimum size of the leaf nodes of the trees (n). The *randomForest* default values for these parameters were 500, a third of the input features, two-thirds of the observations in the training data set and 5, respectively. These defaults were used as the starting points for a grid search procedure that sought to optimise the model hyperparameters.

Table 28: Random forest hyperparameter grid

| Hyper- parameter | Option 1 | Option 2 | Option 3 | Option 4 | Option 5 |
|------------------|----------------------------------|----------------------------------|----------------------------------|----------------------------------|----------|
| B | 500 | 550 | 600 | 650 | 700 |
| K | $\frac{1}{6}$ X No. Features | $\frac{1}{4}$ X No. Features | $\frac{1}{3}$ X No. Features | N/A | N/A |
| L | $\frac{1}{2}$ X No. Observations | $\frac{2}{3}$ X No. Observations | $\frac{3}{4}$ X No. Observations | $\frac{4}{5}$ X No. Observations | N/A |
| n | 3 | 5 | 7 | 9 | N/A |

The search space for the hyperparameter grid search algorithm is provided in Table 28. The number of trees to generate was allowed to range upward from the default value. This was done because *ranger*'s default value for this parameter prioritised execution time of the algorithm. By allowing this parameter to increase upward from the default, the random forest algorithm was provided with a larger search space to optimise this hyperparameter, at the cost of having longer model training times, which was an acceptable downside in this situation. In a similar fashion, the remainder of the hyperparameter grid was created to provide a hyperparameter search space that is wider than the default parameters.

4.2.4.2 Hyperparameter tuning

A combination of grid search and K-fold cross-validation was used to tune the random forest hyperparameters. The grid search was achieved by creating four sequentially nested for-loops, each of which looped through the array of one of the hyperparameters.

The K-fold cross-validation was achieved by creating a final for-loop within the grid-search for-loop structure. The procedure within this loop worked as follows: Similar to the procedure followed during the ANN hyperparameter tuning procedure, the training data set was shuffled and split into three equal-sized parts. Then, the hyperparameters selected by the outer for-loops were passed as parameters to the *ranger* random forest function call, along with two-thirds of the shuffled training data set. The function then used the hyperparameter configuration and the two-thirds training data to build a random forest. The resultant random forest was then validated against the remaining third of the data set. This process was executed three times per hyperparameter configuration, which made it a threefold cross-validation process. Finally, the average threefold cross-validation F1 score was calculated to measure the performance of every hyperparameter configuration. The best performing hyperparameter configurations were selected for each wear type. The final random forest configurations for each of the wheel wear measurements are provided in Table 29.

Table 29: Best performing random forest hyperparameter grid

| Measure | B | K | L | n |
|---------|-----|-----------------------------------|---------------------------------------|---|
| FH | 550 | $\frac{1}{6} \times$ No. Features | $\frac{3}{4} \times$ No. Observations | 9 |
| TD | 550 | $\frac{1}{4} \times$ No. Features | $\frac{2}{3} \times$ No. Observations | 9 |
| HW | 500 | $\frac{1}{6} \times$ No. Features | $\frac{3}{4} \times$ No. Observations | 5 |
| FT | 650 | $\frac{1}{3} \times$ No. Features | $\frac{3}{4} \times$ No. Observations | 5 |
| FS | 550 | $\frac{1}{3} \times$ No. Features | $\frac{1}{3} \times$ No. Observations | 7 |

4.2.4.3 Final model training and evaluation

The final phase of the random forest model development process entailed training the random forest models for each of the wheel wear measurements on the entire train data set, with the hyperparameters set to those in Table 29. As with the ANN and logistic regression models, each model's performance was evaluated based on confusion matrix statistics, ROC curves and the AUC measure.

i) FH model evaluation

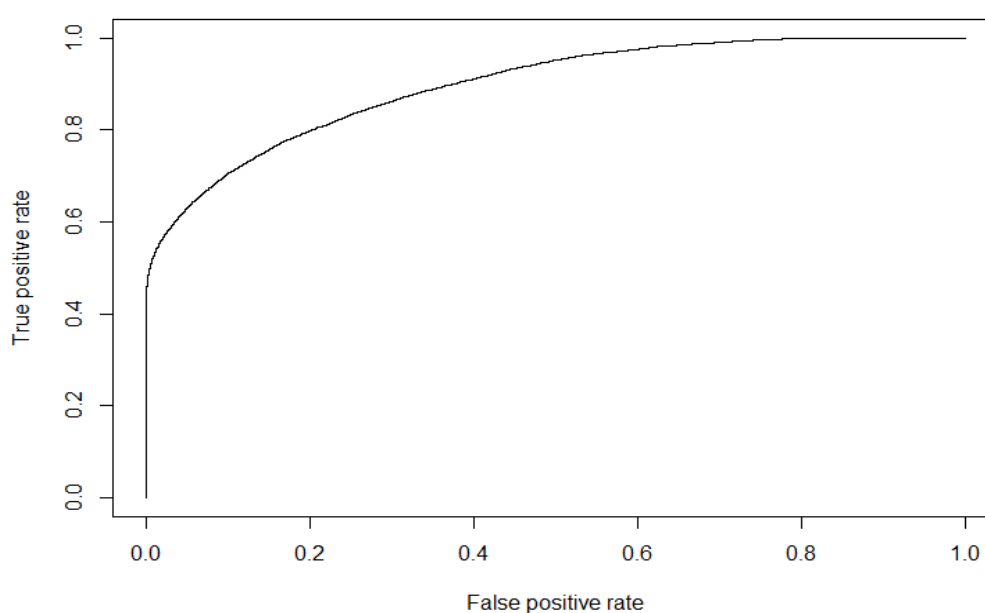
The confusion matrix for the FH wear prognostic random forest model is provided in Table 30. From Table 30, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 31. An ROC curve, which is illustrated in Figure 44, was produced for the ANN model performance on the test set. The associated AUC value was 0.897.

Table 30: Confusion matrix for random forest model of FH wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 4'820 | 5'218 |
| Actual Class = 0 | 183 | 72'681 |

Table 31: Confusion matrix metrics for random forest model of FH wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.480 |
| Specificity | 0.998 |
| Accuracy | 0.935 |
| F1 | 0.964 |

**Figure 44: ROC curve for random forest model of FH wear prognostics**

ii) TD model evaluation

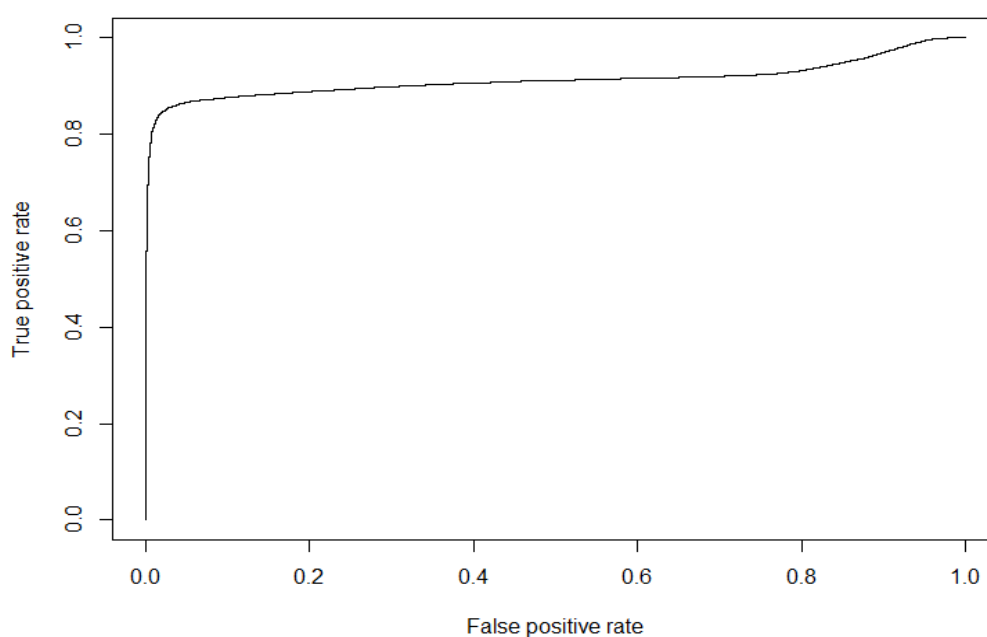
Table 32: Confusion matrix for random forest model of TD wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 41'398 | 0 |
| Actual Class = 0 | 41'492 | 12 |

The confusion matrix for the TD wear prognostic random model is provided in Table 32. From Table 32, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 33. An ROC curve, which is illustrated in Figure 45, was produced for the random forest model performance on the test set. The associated AUC value was 0.913.

Table 33: Confusion matrix metrics for random forest model of TD wear prognostics

| Metric | Value |
|-------------|--------|
| Sensitivity | 1 |
| Specificity | 0.0003 |
| Accuracy | 0.5 |
| F1 | 0.0006 |

**Figure 45: ROC curve for random forest model of TD wear prognostics**

iii) HW model evaluation

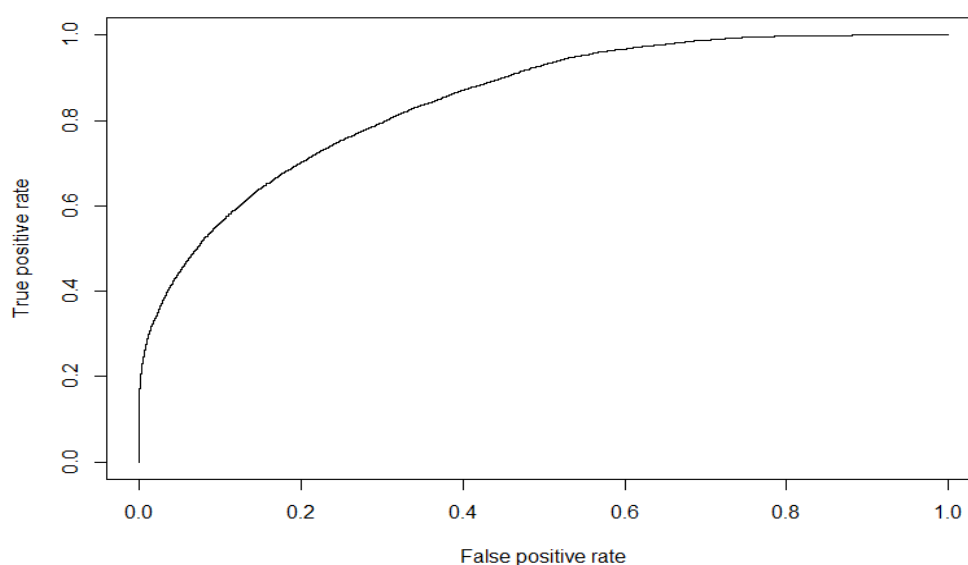
The confusion matrix for the HW wear prognostic random forest model is provided in Table 34. From Table 34, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 35. An ROC curve, which is illustrated in Figure 46, was produced for the random forest model performance on the test set. The associated AUC value was 0.847.

Table 34: Confusion matrix for random forest model of HW wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 1'397 | 5'703 |
| Actual Class = 0 | 106 | 75'696 |

Table 35: Confusion matrix metrics for random forest model of HW wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.196 |
| Specificity | 0.993 |
| Accuracy | 0.925 |
| F1 | 0.960 |

**Figure 46: ROC curve for random forest model of HW wear prognostics**

iv) FS model evaluation

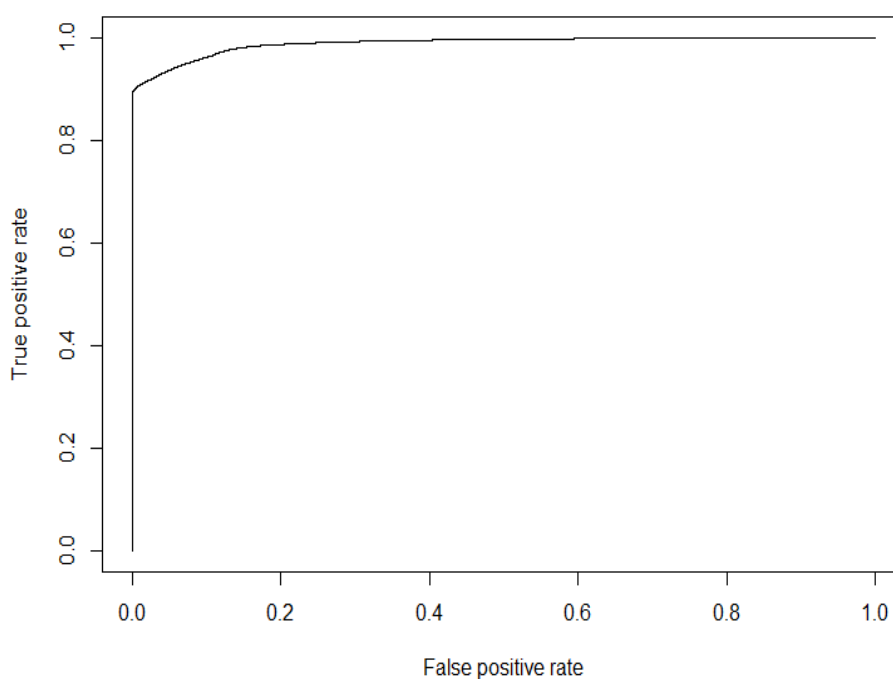
The confusion matrix for the FS wear prognostic random forest model is provided in Table 36. From Table 36, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 37. An ROC curve, which is illustrated in Figure 47, was produced for the ANN model performance on the test set. The associated AUC value was 0.989.

Table 36: Confusion matrix for random forest model of FS wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 52'481 | 3'191 |
| Actual Class = 0 | 1'625 | 25'605 |

Table 37: Confusion matrix metrics for random forest model of FS wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.943 |
| Specificity | 0.940 |
| Accuracy | 0.942 |
| F1 | 0.914 |
| | |

**Figure 47: ROC curve for random forest model of FS wear prognostics**

v) FT model evaluation

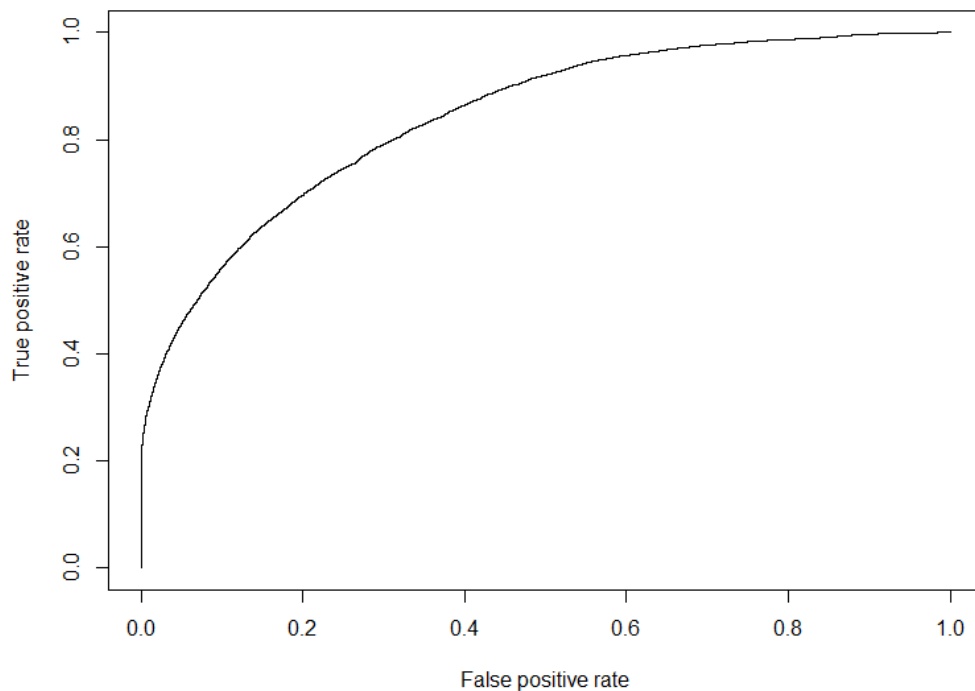
The confusion matrix for the FT wear prognostic random forest model is provided in Table 38. From Table 38, accuracy, sensitivity, specificity, and the F1 measure were calculated. These values are provided in Table 39. An ROC curve, which is illustrated in Figure 48, was produced for the ANN model performance on the test set. The associated AUC value was 0.842.

Table 38: Confusion matrix for random forest model of FT wear prognostics

| | Predicted Class = 1 | Predicted Class = 0 |
|------------------|---------------------|---------------------|
| Actual Class = 1 | 1'537 | 5'869 |
| Actual Class = 0 | 36 | 74'560 |

Table 39: Confusion matrix metrics for random forest model of FT wear prognostics

| Metric | Value |
|-------------|-------|
| Sensitivity | 0.208 |
| Specificity | 1 |
| Accuracy | 0.929 |
| F1 | 0.962 |

**Figure 48: ROC curve for random forest model of FT wear prognostics**

4.2.4.4 Random forest performance summary

The random forest model performed very well when it came to providing FH wear prognostics. The model achieved a high accuracy rate of 93.5% and had an AUC of nearly 0.9, which is high. The model also achieved a moderately high sensitivity rate, indicating that the model was capable of separating the target variable classes. The ROC curve was indicative of a healthy ML model, that curved strongly toward the (0,1) point before bending away toward the (1,1) point.

Random forest had immense difficulty modelling the TD wear. The ROC curve exhibited strange behaviour, with a slight increase in the true positive rate near the (1,1) end of the curve. This indicates that there were two regions of the model's decision threshold where the true positive rate to false positive rate increases, which is unexpected. This further supports the notion that the extrapolated TD data exhibited anomalous behaviour.

Random forest performed well when it came to providing HW prognostics. The model achieved an accuracy rate of 92.5% and had an AUC of 0.847, which is quite high. The ROC curve was also indicative of a healthy and well behaved model, similar to the FH case.

Just like the ANN model, the random forest did extremely well when it came to providing FS wear prognostics. The ROC curve reached close to the (0,1) point and had an AUC of 0.989, which is extremely high. This indicates that the model was capable of separating the target variable classes. To this end, the model achieved an accuracy rate of 94.2%.

Finally, random forest performed well when it came to FT wear. The model attained an accuracy rate of 92.9% and had a high AUC of 0.842. The model had a healthy ROC curve which tended toward the (0,1) point before bending away toward the (1,1) point, indicating that the model was capable of separating the target variable classes.

4.3 Chapter summary

In this chapter the processes followed to build the three ML models for this project are described. The first section describes how the wheel wear measurement data set was ingested and processed in R. Then it describes how the data was formatted and how the outliers in the data set were dealt with. After that it describes how domain-specific knowledge was implemented to engineer features that were added to the data set. The chapter concludes with a description of how the logistic regression, ANN and random forest ML were developed and provides the performance results of the final ML models for each ML model and wheel wear type.

5 Final model selection

In this chapter, the performance of the ML models developed in Chapter 4 will be reviewed to identify the best performing model for each wheel wear type. The chapter starts off with a description of how the various ML model performance metrics were combined to create a single score that was used to select the best suited model for each wheel wear type. The chapter continues with a report of how each model performed, based on the aforementioned score.

5.1 Model score combination

In Chapter 4, various performance metrics for each of the developed ML models are reported for each of the five types of railway wheel wear. However, there still remains the issue of selecting the winning model for each of the wheel wear types. To this end, a method of combining the performance metrics was devised to provide a single metric that was used to rank the models for each wheel wear type.

The difficulty that lead to the development of this combined performance metric stemmed from the high degree of class imbalance in the target variables. The target variable count, grouped by class, is provided in Table 40. As one can see, for most of the wear measurements, a vast majority of the observations fell in the '0' target variable class. This imbalance makes the model metrics difficult to interpret; the reasons for this are discussed in Section 3.2.5.2. To reiterate, a large class imbalance in the output variable of a binary classification problem reduces the merit of many model performance metrics. This is especially true for metrics that only report on the rate of accurate predictions, because the class imbalance will artificially increase such performance metrics. The metrics produced by confusion matrices are particularly susceptible to this phenomenon, because it mainly reports on various accuracy rates of a model. A combined wheel wear measurement was therefore formulated to provide a single measure to rank the ML models, that takes the effect of binary target variable class imbalance into account.

The following equation was used to combine the wheel wear measurements:

$$CS = \frac{(2 \cdot Sens + Spec + 3 \cdot AUC + 2 \cdot F1)}{8} \quad (15)$$

In Eq. 15, CS is a normalised weighted score of the four model performance metrics' sensitivity, specificity, F1 and AUC. The reasoning behind the weights of the model metrics was as follows: specificity had the lowest weight because of the bias toward the negative class, as shown in Table 40. This make it less of an achievement if a model performs well in predicting the negative classes. Sensitivity and the F1 score each received a weight of 2. For sensitivity, this was done to reflect the fact that it is quite an achievement if a model does well in predicting the positive class of the target variable, again due to the target class imbalance shown in Table 40. The F1 score was also allocated with a weight of 2 because it is a balanced score between sensitivity and specificity, that is not severely affected by target variable class imbalance. AUC was assigned the highest weight of 3, because AUC was the metric that was the most target variable class frequency agnostic among the model performance metrics. In other words, AUC was the metric that was least affected by the class imbalance of the binary output variable, which is why this measurement was attributed with the largest weight in the CS calculation.

Table 40: Wheel wear measurement type target variable class counts

| Wear Measurement | Count (Class = 0) | Count (Class = 1) |
|-------------------------|--------------------------|--------------------------|
| FH | 72'864 | 10'038 |
| TD | 41'504 | 41'398 |
| HW | 75'802 | 7'100 |
| FS | 27'230 | 55'672 |
| FT | 75'496 | 7'406 |

5.2 Final model scores and selection

The combined score of the models for each of the wear measurement types are provided in Table 41 to Table 45. The winning model type for each wheel wear measurement is listed in Table 46. The results show that logistic regression performed the poorest of the three ML model types, and was not the best performing model type for any of the wheel wear measurements. ANN performed very well when it came to TD measurement prognostics, and was the best performing model type for the TD measurement type. Of the three model types, random forest was the model type that achieved the highest score for the most wheel measurement types. Random forest was the best performing model type for all the wheel wear measurements, save for TD. This could possibly be attributed to the fact that, of the three model types, random forest is least affected by outliers and missing data.

Table 41: Combined model scores for FH

| Model Type | CS |
|---------------------|-----------|
| Logistic Regression | 0.755 |
| ANN | 0.816 |
| Random Forest | 0.822 |

Table 42: Combined model scores for TD

| Model Type | CS |
|---------------------|-----------|
| Logistic Regression | 0.959 |
| ANN | 0.964 |
| Random Forest | 0.593 |

Table 43: Combined model scores for HW

| Model Type | CS |
|---------------------|-----------|
| Logistic Regression | 0.67 |
| ANN | 0.711 |
| Random Forest | 0.731 |

Table 44: Combined model scores for FS

| Model Type | CS |
|---------------------|-----------|
| Logistic Regression | 0.608 |
| ANN | 0.933 |
| Random Forest | 0.953 |

Table 45: Combined model scores for FT

| Model Type | CS |
|---------------------|-----------|
| Logistic Regression | 0.678 |
| ANN | 0.71 |
| Random Forest | 0.733 |

Table 46: Combined model scores for FT

| Wear Measurement | Highest Scoring Model Type |
|-------------------------|-----------------------------------|
| FH | Random Forest |
| TD | ANN |
| HW | Random Forest |
| FS | Random Forest |
| FT | Random Forest |

5.3 Chapter summary

In this chapter, a method for combining the model scores reported in Chapter 4 into a single model score is described. The chapter continues with a report of this score for each model type and for each wheel measurement type. The chapter concludes by reporting that logistic regression was not the best performing model type for any of the wheel wear measurements. ANN was the best performing model for TD measurements and random forest was the best performing model for FH, HW, FS and TD. This was attributed to the fact that random forest is such a robust model that it is well suited for handling outliers and missing data.

6 Conclusion

The objectives of this research project were, firstly, to build ML models that are capable of providing prognostics on the condition of Metrorail's motor coach wheels, and secondly, to evaluate the predictive performance of the models to ascertain their generalisability. To this end, three ML model types were developed to provide prognostics for five railway wheel wear types. The models were developed to provide a binary prognostic, which indicates if the wheel would have passed the decommissioning threshold for a given wheel wear measurement given an array of input features.

The three ML model types that were compared were logistic regression, artificial neural networks and random forest models. Each model type was developed using a grid search method combined with K-fold cross-validation. The fact that K-fold cross-validation was used in the model development process insured that the results of the ML models were generalisable. This, therefore, ensured that the latter of the main goals for this research project was achieved. K-fold cross-validation allowed for the best performing model to be selected outright from the list of developed models, for each of the wheel wear measurement types.

In terms of the former of the main goals listed, it is very difficult to express the success of an ML model in subjective terms. Therefore, to convey the degree of success of the developed ML models, the absolute and interpretable confusion matrix metrics accuracy, sensitivity, specificity and the ROC metric AUC, are reported to gauge the success of the models. For flange height wear, hollow wear, flange thickness wear and flange slope wear the best performing model was a random forest model, the model development details of which are described in Section 4.2.4. In the case of flange height wear, the model had an accuracy rate of 0.964 with a sensitivity rate of 0.48, a specificity rate of 0.998 and an AUC score of 0.987. The hollow wear model had an accuracy rate of 0.925 with a sensitivity rate of 0.196, a specificity rate of 0.993 and an AUC score of 0.847. The flange slope model had an accuracy rate of 0.942, a sensitivity rate of 0.943, a specificity rate of 0.94 and an AUC score of 0.989. The flange thickness model had an accuracy rate of 0.929, a sensitivity rate of 0.208, a specificity rate of 1 and an AUC score of 0.842. For tread diameter wear, the best performing model was an ANN model, the development details of which are described in subsection ii of Section 4.2.3. The model had an accuracy rate of 0.963, a sensitivity rate of 0.930, a specificity rate of 0.996 and an AUC score of 0.977.

Random forest was deemed to be the best performing model of the three model types, overall. The reason for this is that the random forest models were the best performing models, for all wheel wear measurements, save for tread diameter wear. However, it was determined that tread diameter wear could safely be ignored when it came to selecting the best model type in this research. Firstly, because the TD wear data represented the noisiest, and most irregular data set among the five types of wheel wear for which data was collected. The reason for this was because tread diameter, as measured by the *MinProf* during condition monitoring, is itself a calculated measurement. The *MinProf* estimates a wheel's diameter, based on the measured profile of a wheel over a very narrow area. This extrapolation resulted in a noisy set of tread diameter measurements. Finally, it was advised by Metrorail's rolling stock maintenance management that TD wear should be deprioritised when it came to selecting the best performing model type, overall. This suggestion also stemmed from concerns that Metrorail had regarding the noisiness and inaccuracy of the *MinProf*'s estimates of wheel diameter. For these reasons, the random forest model type was deemed to be the most successful of the three model types that were used to address the specific problem for this research project's case study.

From these metrics, it is clear that the ML models were successful in providing prognostic predictions on the sample data provided by PRASA. This, therefore, illustrates that the former of the two goals of this research project has been achieved. For PRASA, this is the first case where ML techniques were implemented to aid with providing wheel wear prognostics. This represents a major step forward for PRASA and forms part of PRASA's push to move from reactive to preventive maintenance with the help of ML and data analytics.

7 Recommendations

After consideration of the results produced during the completion of this research project, it is highly recommended that PRASA move forward with this research by implementing ML techniques for railway wheel prognostics in a live maintenance implementation environment. It is suggested that this should be done by selecting a sample of Metrorail's motor coach wheels and ascertaining whether the models continue to provide accurate prognostics for these wheels, as compared to routine readings taken by the *MiniProf* during condition monitoring inspections.

The suggestion for how these models ought to be implemented is depicted in Figure 49. The figure depicts an array of data which represents a combination of data that is entered by a Metrorail maintenance technician as well as wheel condition monitoring data that is automatically captured by the *MiniProf*. The data that is entered by the technician will encapsulate the estimated future values for various metrics at the next point in time when a given railway wheel will be brought in for condition monitoring. This information, combined with the data recorded by the *MiniProf*, is fed into the ML models that were developed in this thesis. The models are served as functions on a server, that accepts arrays of wheel monitoring data as an argument. The models will then use the ingested data array to produce a binary prediction, for each of the wheel wear types, of whether a specific form of wheel wear has exceeded its allowed extent.

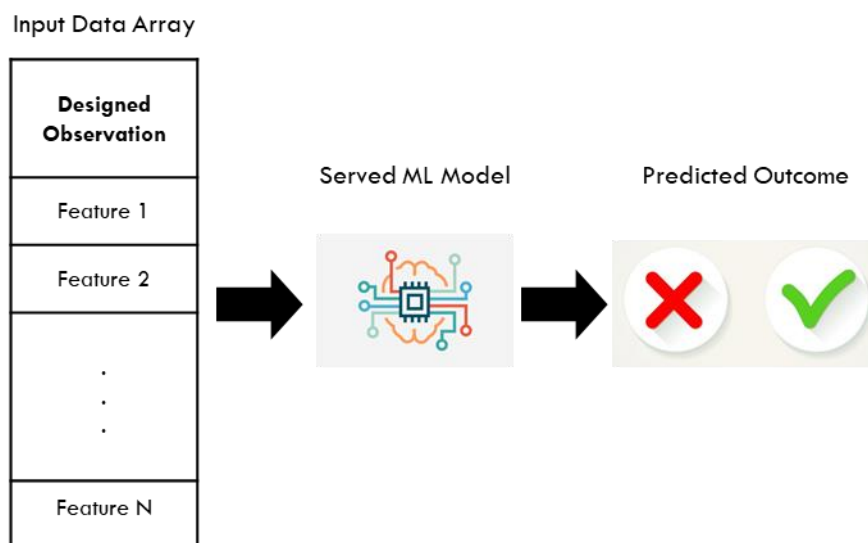


Figure 49: Suggested framework for ML model implementation

The suggestions for future research stemmed from the limitations of the research presented in this thesis. These limitations were, specifically, the fact that the models that were developed for this research were developed on a specific data set provided by Metrorail. Furthermore, for the random forest and ANN models, the range of hyperparameters that were considered for optimisation were limited. Finally, the research only considered three model types, namely, logistic regression, ANN and random forest. These limitations provided opportunity for expansion of the research presented in this thesis.

Firstly, Metrorail and PRASA possess immense volumes of data related to various aspects of the organisation, such as consumer behaviour related data, workforce efficiency related data and asset condition related data to name a few. If these data sources can be consolidated into a more effective and coherent whole, it can be used to build ML models that are potentially more performant than the ones developed during the course of this research. Furthermore, if the range of hyperparameters that are considered for model optimisation can be expanded then the model performance can potentially be improved. Finally, there exists a myriad of ML model types and possible model configurations which represents a large field of inquiry that can be explored in future research to produce ML models that are capable of aiding Metrorail and PRASA in numerous aspects of their endeavours.

References

- [1] (FHWA) Federal Highway Administration, "Status of the nation's highways, bridges, and transit: 2002 conditions and performance report.," 2003, 2003.
- [2] T. Litman, "Rail transit in America: a comprehensive evaluation of benefits.," trid.trb.org, 2015.
- [3] L. Redman, M. Friman, T. Gärling, and T. Hartig, "Quality attributes of public transport that attract car users: A research review," *Transport Policy*, vol. 25, pp. 119–127, Jan. 2013.
- [4] D. P. Solomatine and A. Ostfeld, "Data-driven modelling: some past experiences and new approaches," *Journal of Hydroinformatics*, vol. 10, no. 1, pp. 3–22, Jan. 2008.
- [5] S. Salzberg, R. Chandar, H. Ford, S. K. Murthy, and R. White, "Decision trees for automated identification of cosmic-ray hits in Hubble Space Telescope images," *PUBL. ASTRON. SOC. PAC.*, vol. 107, p. 279, Mar. 1995.
- [6] P. Baldi, P. Sadowski, and D. Whiteson, "Searching for exotic particles in high-energy physics with deep learning.," *Nat. Commun.*, vol. 5, p. 4308, Jul. 2014.
- [7] H. Li, D. Parikh, Q. He, B. Qian, Z. Li, D. Fang, and A. Hampapur, "Improving rail network velocity: A machine learning approach to predictive maintenance," *Transportation Research Part C: Emerging Technologies*, vol. 45, pp. 17–26, Aug. 2014.
- [8] "PRASA Rail Profile| Organisational Structure | About Us | PRASA Corporate." [Online]. Available: <https://www.prasa.com/PRASA%20Rail%20Profile.html>. [Accessed: 15-Aug-2018]
- [9] "Welcome to Metrorail." [Online]. Available: <http://www.metrorail.co.za/About.html>. [Accessed: 15-Aug-2018]
- [10] X. Liu, M. R. Saat, and C. P. L. Barkan, "Analysis of causes of major train derailment and their effect on accident rates," *Transportation Research Record*, vol. 2289, no. 1, pp. 154–163, Jan. 2012.
- [11] C. Wei, Q. Xin, W. H. Chung, S. Liu, H. Tam, and S. L. Ho, "Real-Time Train Wheel Condition Monitoring by Fiber Bragg Grating Sensors," *International Journal of Distributed Sensor Networks*, vol. 8, no. 1, p. 409048, Jan. 2012.
- [12] S. Dirks, "Metrorail Motor Coach Wheel Discussion," 07-Sep-2018.
- [13] "South Devon Railway - Railway Wheel Construction." [Online]. Available: <http://www.southdevonrailwayassociation.org/Railway-Wheels.html>. [Accessed: 08-Sep-2018]
- [14] "Train Wheel Set Dimensions - Best Train 2018." [Online]. Available: <http://www.kampungan.co/train-wheel-set-dimensions/>. [Accessed: 12-Sep-2018]
- [15] New South Wales. Office of Transport Safety Investigations, "Derailment of freight train MB520, Pangel, New South Wales, 28 August 2015," Australian Transport Safety Bureau, Rail occurrence investigation RO-2015-015, Jun. 2016.
- [16] I. Y. Shevtsov, V. L. Markine, and C. Esveld, "Design of railway wheel profile taking into account rolling contact fatigue and wear," *Wear*, vol. 265, no. 9–10, pp. 1273–1282, Oct. 2008.
- [17] "Greenwood Engineering - MiniProf - MiniProf BT Wheel," MiniProf BT Wheel. [Online]. Available: <https://www.greenwood.dk/MiniProfbtwheel.php>. [Accessed: 16-Sep-2018]

2018]

- [18] Y. S. Abu-Mostafa, M. Magdon-Ismail, and H. T. Lin, *Learning from data*, vol. 4. New York NY USA: AMLBook, 2012 [Online]. Available: <http://work.caltech.edu/homework/final.pdf>. [Accessed: 07-May-2018]
- [19] "What is maintenance? definition and meaning - BusinessDictionary.com." [Online]. Available: <http://www.businessdictionary.com/definition/maintenance.html>. [Accessed: 03-Jul-2018]
- [20] H. Wang, "A survey of maintenance policies of deteriorating systems," *European Journal of Operational Research*, vol. 139, no. 3, pp. 469–489, Jun. 2002.
- [21] J. Endrenyi, S. Aboresheid, R. N. Allan, G. J. Anders, S. Asgarpour, R. Billinton, N. Chowdhury, E. N. Dialynas, M. Fipper, R. H. Fletcher, C. Grigg, J. McCalley, S. Meliopoulos, T. C. Mielnik, P. Nitu, N. Rau, N. D. Reppen, L. Salvaderi, A. Schneider, and C. Singh, "The present status of maintenance strategies and the impact of maintenance on reliability," *IEEE Trans. Power Syst.*, vol. 16, no. 4, pp. 638–646, 2001.
- [22] "System reliability [Article about system reliability by The Free Dictionary." [Online]. Available: <http://encyclopedia2.thefreedictionary.com/system+reliability>. [Accessed: 04-Feb-2018]
- [23] "asset |Definition of asset in English by Oxford Dictionaries." [Online]. Available: <https://en.oxforddictionaries.com/definition/asset>. [Accessed: 22-Mar-2018]
- [24] J. E. Amadi-Echendu, R. Willett, K. Brown, T. Hope, J. Lee, J. Mathew, N. Vyas, and B.-S. Yang, "What is engineering asset management?," in *Definitions, concepts and scope of engineering asset management*, vol. 1, J. E. Amadi-Echendu, K. Brown, R. Willett, and J. Mathew, Eds. London: Springer London, 2010, pp. 3–16.
- [25] G. Sullivan, R. Pugh, A. P. Melendez, and W. D. Hunt, "Operations & Maintenance Best Practices - A Guide to Achieving Operational Efficiency (Release 3)," Pacific Northwest National Laboratory (PNNL), Richland, WA (United States), Aug. 2010.
- [28] R. Ahmad and S. Kamaruddin, "An overview of time-based and condition-based maintenance in industrial application," *Computers & Industrial Engineering*, vol. 63, no. 1, pp. 135–149, Aug. 2012.
- [29] H. Pham and H. Wang, "Imperfect maintenance," *European Journal of Operational Research*, vol. 94, no. 3, pp. 425–438, Nov. 1996.
- [30] B. Wu, Z. Tian, and M. Chen, "Condition-based Maintenance Optimization Using Neural Network-based Health Condition Prediction," *Qual. Reliab. Engng. Int.*, vol. 29, no. 8, pp. 1151–1163, Dec. 2013.
- [31] "diagnosis |Definition of diagnosis in English by Oxford Dictionaries." [Online]. Available: <https://en.oxforddictionaries.com/definition/diagnosis>. [Accessed: 07-Apr-2018]
- [32] "prognosis |Definition of prognosis in English by Oxford Dictionaries." [Online]. Available: <https://en.oxforddictionaries.com/definition/prognosis>. [Accessed: 12-Apr-2018]
- [33] W. W. Tiddens, A. J. J. Braaksma, and T. Tinga, "The adoption of prognostic technologies in maintenance decision making: A multiple case study," *Procedia CIRP*, vol. 38, pp. 171–176, 2015.
- [34] H. M. Elattar, H. K. Elminir, and A. M. Riad, "Prognostics: a literature review," *Complex Intell. Syst.*, vol. 2, no. 2, pp. 125–154, Jun. 2016.
- [35] R. C. Yam, P. W. Tse, L. Li, and P. Tu, "Intelligent predictive decision support system for condition-based maintenance," *The International Journal of Advanced Manufacturing Technology*, vol. 17, no. 5, pp. 383–391, 2001 [Online]. Available: <https://link.springer.com/article/10.1007/s001700170173>. [Accessed: 04-Feb-2018]
- [36] T. M. (Tom M. Mitchell, *Machine Learning*. New York: McGraw-Hill, 1997.

- [37] S. Marsland, Machine learning: an algorithmic perspective. CRC press, 2015.
- [38] G. James, D. Witten, T. Hastie, and R. Tibshirani, An Introduction to Statistical Learning, vol. 103. New York, NY: Springer New York, 2013.
- [39] S. A. Czepiel, "Maximum likelihood estimation of logistic regression models: theory and implementation.," Available at czep.net/stat/mlelr.pdf, 2002 [Online]. Available: <http://www.saedsayad.com/docs/mlelr.pdf>. [Accessed: 10-Jun-2018]
- [40] S. Agatonovic-Kustrin and R. Beresford, "Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research.," J. Pharm. Biomed. Anal., vol. 22, no. 5, pp. 717–727, Jun. 2000.
- [41] I. A. Basheer and M. Hajmeer, "Artificial neural networks: fundamentals, computing, design, and application.," J. Microbiol. Methods, vol. 43, no. 1, pp. 3–31, Dec. 2000.
- [42] A. Lemmens and C. Croux, "Bagging and boosting classification trees to predict churn," Journal of Marketing Research, vol. 43, no. 2, pp. 276–286, May 2006.
- [43] T. Hastie, R. Tibshirani, and J. H. (Jerome H. Friedman, The elements of statistical learning: Data mining, inference, and prediction, 2nd ed. New York, NY: Springer, 2009.
- [44] C.-W. Hsu, C.-C. Chang, and C.-J. Lin, "A practical guide to support vector classification," Department of Computer Science, National Taiwan University, Technical, Apr. 2010.
- [45] "View and set current colormap - MATLAB colormap." [Online]. Available: <https://www.mathworks.com/help/matlab/ref/colormap.html>. [Accessed: 21-Jul-2018]
- [46] A. P. Bradley, "The use of the area under the ROC curve in the evaluation of machine learning algorithms. ," Pattern Recognit, vol. 30, no. 7, pp. 1145–1159, 1997.
- [47] K. Barraclough, "Diagnosis: shifting the ROC curve," Br. J. Gen. Pract., vol. 62, no. 602, pp. 452–453, Sep. 2012.
- [48] I. H. (Ian H. Witten, E. Frank, and M. A. Hall, Data mining: Practical machine learning tools and techniques, 3rd ed. Burlington, MA: Morgan Kaufmann, 2011.
- [49] M. Bowles, Machine Learning in Python: Essential techniques for predictive analysis. Indianapolis, IN: John Wiley & Sons, Inc., 2015.
- [50] "Gradient Descent Algorithm and Its Variants – Towards Data Science." [Online]. Available: <https://towardsdatascience.com/gradient-descent-algorithm-and-its-variants-10f652806a3?gi=8cfd7ab31d6b>. [Accessed: 10-Nov-2018]

Appendix A

Data Preparation and Modelling Code Sample

```
#cleaning working environment
rm(list = ls())
gc()
memory.limit(size=1000000)
#loading library to read excel sheet into dataframe
library(xlsx)
library(dplyr)

#load wheel data into dataframe
wl_data = read.csv(file = 'c:/Users/jduplessis/Documents/PRASA DATA/data.csv')

#checking data table entries
head(wl_data)

#finding indices of dot spaced field names
grep("\\.", colnames(wl_data))

##changing dot spaced field names to underscore spaced field names
colnames(wl_data)[9:10] = c('axle_number', 'wheel_id')

##renaming measurement fields
colnames(wl_data)[c(12,13,14,15,16)] = c('FH','TD','HW','FT','FS')

#selecting columns that can safely be dropped
drop_columns = c('Filename','OperatorName','WheelT','Dup','Field16')

#dropping selected columns
wl_data = wl_data[, -which(colnames(wl_data) %in% drop_columns)]

#produce initial summary of data
summary(wl_data)

save.image(file='C:/Users/fkimokoti.000/Desktop/PRASA DATA/johan_workspace.RData')

#Recoding Stock values
wl_data[wl_data$Stock == '10M5t','Stock'] = '10M5T'
wl_data[wl_data$Stock == '5M2At','Stock'] = '5M2AT'

#Date and Time is in excel serial format
##converting serial date to dd-mm-yyyy date
```

```

wl_data$Date_formatted = as.Date(wl_data$Date, origin = "1899-12-30")

##checking formatted Date field
summary(wl_data$Date_formatted)

##converting time to hh seeing as train should not be operating within minutes of being
maintained
wl_data$Time_hour_modulo = (wl_data$Time*86400) %% (60*60)
wl_data$Time_hour = ((wl_data$Time*86400) - wl_data$Time_hour_modulo)/(60*60)

##spot checking records with missing Time field for misalignment
wl_data[which(is.na(wl_data$Time))[1:10],]
##records appear to be in line

##substituting missing Time with median time (to avoid minutes created by mean)
summary(wl_data$Time_hour)
wl_data[is.na(wl_data$Time_hour),'Time_hour'] = 11

##creating singular, formatted date time field
wl_data$Time_hour_formatted = ifelse(wl_data$Time_hour < 10,
paste0('0',wl_data$Time_hour,':','00'), paste0(wl_data$Time_hour,':','00'))
wl_data$Datetime = paste(wl_data$Date_formatted, wl_data$Time_hour_formatted, sep = ' ')

##converting date time field to datetime format
wl_data$Datetime = as.POSIXct(wl_data$Datetime, format = '%Y-%m-%d %H:%M')

#dropping off helper fields
wl_data = wl_data[,-
which(colnames(wl_data)%in%c('Time','Date','Date_formatted','Time_hour_modulo','Time_hour',
'Time_hour_formatted'))]

#dummying out categorical variables

##passing through categorical variables and transforming into wide format
library(dummies)

###stock
stock = dummy(wl_data$Stock,sep = '_')
stock = data.frame(stock)

###set
SetNo = dummy(wl_data$SetNo,sep = '_')
SetNo = data.frame(SetNo)

###coach
CoachNo = dummy(wl_data$CoachNo,sep = '_')
CoachNo = data.frame(CoachNo)

```



```

####bogie nubmer
BogieNo = dummy(wl_data$BogieNo,sep = '_')
BogieNo = data.frame(BogieNo)

####axle nubmer
axle_number = dummy(wl_data$axle_number,sep = '_')
axle_number = data.frame(axle_number)

####wheel id
wheel_id = dummy(wl_data$wheel_id,sep = '_')
wheel_id = data.frame(wheel_id)

##attaching dummied variables to placeholder dataframe
wl_data = cbind(wl_data, stock, BogieNo, axle_number, wheel_id)

rm(stock, BogieNo, axle_number, wheel_id)

#converting dummied variables to factors
wl_data[,which(colnames(wl_data) == 'Stock_10M2'):ncol(wl_data)] =
lapply(wl_data[,which(colnames(wl_data) == 'Stock_10M2'):ncol(wl_data)], as.factor)
summary(wl_data)

#flaggin erroneous measurements
wl_data$TD_flag = 0
wl_data[wl_data$TD < 1 | wl_data$TD > 1999 | is.na(wl_data$TD),'TD_flag'] = 1

wl_data$FH_flag = 0
wl_data[is.na(wl_data$FH),'FH_flag'] = 1

wl_data$HW_flag = 0
wl_data[is.na(wl_data$HW),'HW_flag'] = 1

wl_data$FT_flag = 0
wl_data[is.na(wl_data$FT),'FT_flag'] = 1

wl_data$FS_flag = 0
wl_data[is.na(wl_data$FS),'FS_flag'] = 1

wl_data$Faulty_count = wl_data$TD_flag + wl_data$FH_flag + wl_data$HW_flag +
wl_data$FT_flag + wl_data$FS_flag

wl_data[,c('TD_flag','FH_flag','FT_flag','FS_flag','Faulty_count','HW_flag')] =
lapply(wl_data[,c('TD_flag','FH_flag','FT_flag','FS_flag','Faulty_count','HW_flag')],as.factor)

summary(wl_data)

```

```

# write.table(wl_data, file = 'c:/Users/jduplessis/Documents/PRASA DATA/wheel_data.txt', sep =
'|', col.names = T, row.names = F, quote = F)

#creating unique wheel instance ID's
##first, sampling 10 wheel instances that had more than 20 measurements
library(sqldf)
wheel_measure_counts = sqldf("select count(*) as measurements, Stock, SetNo, CoachNo,
BogieNo, axle_number, wheel_id
from wl_data
group by Stock, SetNo, CoachNo, BogieNo, axle_number, wheel_id
order by measurements desc")
summary(wheel_measure_counts$measurements)
wheel_measure_counts20 = wheel_measure_counts[wheel_measure_counts$measurements
>= 20,]

set.seed(1234)
datasample_pos =
wheel_measure_counts20[sample(1:nrow(wheel_measure_counts20),10,replace = F),]
datasample = sqldf("select
b.ID, b.Stock, b.SetNo, b.CoachNo, b.BogieNo, b.axle_number,
b.wheel_id, b.FH, b.Datetime
from
datasample_pos a
inner join
wl_data b
on a.Stock = b.Stock
and a.SetNO = b.SetNo
and a.CoachNo = b.CoachNo
and a.BogieNo = b.BogieNo
and a.axle_number = b.axle_number
and a.wheel_id = b.wheel_id
order by b.ID, b.Stock, b.SetNo, b.CoachNo, b.BogieNo, b.axle_number")

datasample =
datasample[order(datasample[,2],datasample[,3],datasample[,4],datasample[,5],datasample[,6],
datasample[,7],datasample[,9], decreasing = F),]

# write.table(datasample, file = 'c:/Users/jduplessis/Documents/PRASA
DATA/sampleddata.txt', sep = '|', col.names = T, row.names = F, quote = F)

##plotting single sample data
library(ggplot2)

##plotting uncleaned sample
single_sample = read.xlsx(file = 'C:/Users/jduplessis/Documents/PRASA
DATA/single_sample.xlsx', sheetIndex = 1)

```

```

single_sample_no_instance1 = single_sample[,c('Date','wheel_instance_1')]
single_sample_no_instance1 =
single_sample_no_instance1[complete.cases(single_sample_no_instance1),]
single_sample_no_instance2 = single_sample[,c('Date','wheel_instance_2')]
single_sample_no_instance2 =
single_sample_no_instance2[complete.cases(single_sample_no_instance2),]
colnames(single_sample_no_instance1) = c('date','FH')
colnames(single_sample_no_instance2) = c('date','FH')
single_sample_no_instance = rbind(single_sample_no_instance1,single_sample_no_instance2)

```

```

single_sample_no_instance$date = as.Date(single_sample_no_instance$date)

```

```

plot = ggplot(data = single_sample_no_instance, aes(x = date, y = FH)) +
  geom_line() +
  ylim(10, 40) +
  scale_x_date(date_labels = '%Y-%m', date_breaks = '2 month')+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90,vjust=0.5))+
  xlab('Date (yyyy-mm)')+
  ylab('Flange Height (mm)')
plot

```

```

##plotting uncleaned sample

```

```

single_sample_nosplit = read.xlsx(file = 'C:/Users/jduplessis/Documents/PRASA
DATA/single_sample.xlsx', sheetIndex = 2)

```

```

single_sample_nosplit$date = as.Date(single_sample_nosplit$date)

```

```

plot2 = ggplot(data = single_sample_nosplit, aes(x = date, y = FH_fixed)) +
  geom_line() +
  ylim(10, 40) +
  scale_x_date(date_labels = '%Y-%m', date_breaks = '2 month')+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90,vjust=0.5))+
  xlab('Date (yyyy-mm)')+
  ylab('Flange Height (mm)')
plot2

```

```

##plotting cleaned after split

```

```

single_sample_instance1 = single_sample[,c('Date','wheel_instance_1_fixed')]
single_sample_instance1$instance = 'Wheel Instance 1'
single_sample_instance1 =
single_sample_instance1[complete.cases(single_sample_instance1),]
single_sample_instance2 = single_sample[,c('Date','wheel_instance_2_fixed')]
single_sample_instance2$instance = 'Wheel instance 2'
single_sample_instance2 =
single_sample_instance2[complete.cases(single_sample_instance2),]
colnames(single_sample_instance1) = c('date','FH','Wheel Instance')

```

```

colnames(single_sample_instance2) = c('date','FH','Wheel Instance')
single_sample_instance = rbind(single_sample_instance1,single_sample_instance2)

single_sample_instance$date = as.Date(single_sample_instance$date)

plot3 = ggplot(data = single_sample_instance, aes(x = date, y = FH, group=`Wheel Instance`,
colour = `Wheel Instance`)) +
  geom_line() +
  ylim(10, 40) +
  scale_x_date(date_labels = '%Y-%m', date_breaks = '2 month')+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90,vjust=0.5))+
  xlab('Date (yyyy-mm)')+
  ylab('Flange Height (mm)')
plot3

```

```

##sample data was cloned and instances identified in excel, readin in the excel sheet
library(xlsx)
datasample = read.xlsx(file = 'c:/Users/jduplessis/Documents/PRASA
DATA/sampleddata.xlsx',sheetName = 'raw data')
datasample$Measurediff = 0
datasample[1,'Measurediff'] = NA
datasample$Datediff = 0
datasample[1,'Datediff'] = NA

##creating subsequent measurement difference variable and subsequent measurment time
difference for wheel instances
for(i in 2:nrow(datasample)){
  if(datasample[i,1] == datasample[i-1,1] & datasample[i,2] == datasample[i-1,2] &
datasample[i,3] == datasample[i-1,3] & datasample[i,4] == datasample[i-1,4] & datasample[i,5]
== datasample[i-1,5] & datasample[i,6] == datasample[i-1,6] & datasample[i,9] == datasample[i-
1,9]){
    datasample[i,'Measurediff'] = datasample[i,'FH'] - datasample[i-1,'FH']
    datasample[i,'Datediff'] = difftime(datasample[i,'Datetime'],datasample[i-1,'Datetime'], units =
'weeks')
  }else{
    datasample[i,'Measurediff'] = NA
    datasample[i,'Datediff'] = NA
  }
  print(i)
}

```

```

##normalising subsequent FH measurement differences to difference/week
datasample$Measurediff_per_week = datasample$Measurediff/datasample$Datediff

```

```

##overwriting previous sample dataset
#### write.table(datasample, file = 'c:/Users/jduplessis/Documents/PRASA
DATA/sampleddata.txt',sep = '|', col.names = T, row.names = F, quote = F)

```

```

##reading in cleaned data
datasample = read.csv(file = 'c:/Users/jduplessis/Documents/PRASA DATA/sampleddata.txt',sep
= '|')

##creating box plot of average weekly FH wear differnce
fh_diff_stats = data.frame('stats' = c(-0.128,0.005,0.035,0.076,0.200), 'x' = rep(0,5))

box = ggplot(data = datasample, aes(y = Measurediff_per_week))+
  geom_boxplot(coef = 2)+
  theme_bw()+
  theme(axis.text.x=element_blank(),axis.ticks.x=element_blank())+
  # geom_text(data = fh_diff_stats, aes(x = 0.41,y = stats, label = stats), vjust = 0.4, size = 3)+
  ylab('FH Wear per Week (mm)')+
  xlab("")+
  scale_y_continuous(breaks=c(-0.250,-0.128,0.005,0.035,0.076,0.200, 0.250, 0.500))
box

##creating box plot of negative FH wear differnce instances
fh_diff_stats = data.frame('stats' = c(-0.128,0.005,0.035,0.076,0.200), 'x' = rep(0,5))

box = ggplot(data = datasample[datasample$Measurediff <0,], aes(y = Measurediff))+
  geom_boxplot(coef = 1.5)+
  theme_bw()+
  theme(axis.text.x=element_blank(),axis.ticks.x=element_blank())+
  # geom_text(data = fh_diff_stats, aes(x = 0.41,y = stats, label = stats), vjust = 0.4, size = 3)+
  ylab('Consecutive FH Measurement Difference (mm)')+
  xlab("")+
  scale_y_continuous(breaks=c(-0.4170,-0.18785,-0.05980,-0.02832,0.0005))
box

##implementing wheel instance flagging and FH cleaning algorithm
##grouping/sorting wl_data into unique stock, SetNo, CoachNo, BogieNo, axle_number,
wheel_id group sets
wl_data = sqldf("select *
                from wl_data
                order by
                Stock, SetNo, CoachNo, BogieNo, axle_number, wheel_id, Datetime asc")

wl_data$FH_change = NA
wl_data$Datetime_change = NA
wl_data$FH_weekly_change = NA

wl_data$ID = 1
wl_data = unique(wl_data)

##calculating FH change

for(i in 2:nrow(wl_data)){

```

```

if(wl_data[i,2] == wl_data[i-1,2] & wl_data[i,3] == wl_data[i-1,3] & wl_data[i,4] == wl_data[i-1,4]
& wl_data[i,5] == wl_data[i-1,5] & wl_data[i,6] == wl_data[i-1,6] & wl_data[i,7] == wl_data[i-1,7]){
  wl_data[i,'FH_change'] = wl_data[i,'FH'] - wl_data[i-1,'FH']
  wl_data[i,'Datetime_change'] = difftime(wl_data[i,'Datetime'],wl_data[i-1,'Datetime'], units =
'weeks')
  wl_data[i,'FH_weekly_change'] = wl_data[i,'FH_change']/wl_data[i,'Datetime_change']
}
print(i)
}

```

```

##cleaning up same date FH change (-Inf, Inf)
wl_data$inf_flag = 0
wl_data$FH_weekly_change2 = wl_data$FH_weekly_change
for(i in 1:nrow(wl_data)){
  if((wl_data[i,'FH_weekly_change2'] == Inf | wl_data[i,'FH_weekly_change2'] == -Inf) &
!is.na(wl_data[i,'FH_weekly_change2'])){
    wl_data[i,'inf_flag'] = 1
    wl_data[i-1,'FH_weekly_change2'] = mean(wl_data[i-
1,'FH_change'],wl_data[i,'FH_change'])/wl_data[i-1,'Datetime_change']
    wl_data[i,'FH_weekly_change2'] = NA
    print(i)
  }
}

```

```

wl_data$measurement_id = 1
##creating measurment ids
for(i in 2:nrow(wl_data)){
  if(wl_data[i,2] == wl_data[i-1,2] & wl_data[i,3] == wl_data[i-1,3] & wl_data[i,4] == wl_data[i-1,4]
& wl_data[i,5] == wl_data[i-1,5] & wl_data[i,6] == wl_data[i-1,6] & wl_data[i,7] == wl_data[i-1,7]){
    wl_data[i,'measurement_id'] = wl_data[i-1,'measurement_id']+1
    print( wl_data[i,'measurement_id'])
  }
}

```

```

###flagging and dropping stock, SetNo, CoachNo, BogieNo, axle_number, wheel_id group sets
with less than 10 measurements
wl_data$less10_drop_flag = 0
for(i in 3:nrow(wl_data)){
  if(wl_data[i,'measurement_id'] == 1){
    ten_check = wl_data[i-1,'measurement_id']
    if(ten_check < 10){
      wl_data[(i-ten_check):(i-1),'less10_drop_flag'] = 1
      print(ten_check)
    }
  }
}

```

```

wl_data2 = wl_data[wl_data$less10_drop_flag != 1,]

```

```

wl_data2 = wl_data2[wl_data2$FH >= 20,]
##applying outlier detection logic
wl_data2$wheel_change_flag = 0
wl_data2$FH_fixed = wl_data2$FH
wl_data2 = wl_data2[!is.na(wl_data2$FH),]

for(i in 2:nrow(wl_data2)){
  if(wl_data2[i,2] == wl_data2[i-1,2] & wl_data2[i,3] == wl_data2[i-1,3] & wl_data2[i,4] ==
wl_data2[i-1,4] & wl_data2[i,5] == wl_data2[i-1,5] & wl_data2[i,6] == wl_data2[i-1,6] &
wl_data2[i,7] == wl_data2[i-1,7] & wl_data2[i,'measurement_id'] != 1){

    wl_data2[i,'FH_change'] = wl_data2[i,'FH'] - wl_data2[i-1,'FH_fixed']
    wl_data2[i,'Datetime_change'] = difftime(wl_data2[i,'Datetime'],wl_data2[i-1,'Datetime'], units =
'weeks')
    wl_data2[i,'FH_weekly_change'] = wl_data2[i,'FH_change']/wl_data2[i,'Datetime_change']

    if(wl_data2[i,'FH_weekly_change'] > 0){
      if(wl_data2[i,'FH_weekly_change'] <= 0.2){
        wl_data2[i,'FH_fixed'] = wl_data2[i,'FH']
      }else if(wl_data2[i,'FH_weekly_change'] > 0.2){
        if(wl_data2[i+1,'measurement_id'] != 1){

          wl_data2[i+1,'FH_change'] = wl_data2[i+1,'FH'] - wl_data2[i,'FH']
          wl_data2[i+1,'Datetime_change'] = difftime(wl_data2[i+1,'Datetime'],wl_data2[i,'Datetime'],
units = 'weeks')
          wl_data2[i+1,'FH_weekly_change'] =
wl_data2[i+1,'FH_change']/wl_data2[i+1,'Datetime_change']

          if(wl_data2[i+1,'FH_weekly_change'] <= -0.128){
            wl_data2[i,'FH_fixed'] = mean(wl_data2[i-1,'FH_fixed'],wl_data2[i+1,'FH'])
          }
        }
      }else if(wl_data2[i,'FH_weekly_change'] < 0){
        if(wl_data2[i,'FH_change'] <= -0.417){
          if(wl_data2[i+1,'measurement_id'] != 1){

            wl_data2[i+1,'FH_change'] = wl_data2[i+1,'FH'] - wl_data2[i,'FH']
            wl_data2[i+1,'Datetime_change'] = difftime(wl_data2[i+1,'Datetime'],wl_data2[i,'Datetime'],
units = 'weeks')
            wl_data2[i+1,'FH_weekly_change'] =
wl_data2[i+1,'FH_change']/wl_data2[i+1,'Datetime_change']

            if(wl_data2[i+1,'FH_weekly_change'] >= 0.2){
              wl_data2[i,'FH_fixed'] = wl_data2[i-1,'FH']
            }else if(wl_data2[i+1,'FH_weekly_change'] < 0.2){
              wl_data2[i,'FH_fixed'] = wl_data2[i,'FH']
              wl_data2[i,'wheel_change_flag'] = 1
            }
          }
        }
      }
    }
  }
}

```

```

    }
  }
  }else if(wl_data2[i,'FH_change'] > -0.417){
    wl_data2[i,'FH_fixed'] = wl_data2[i-1,'FH_fixed']
  }
}
}
print(i)
}

```

##using wheel change flag to create wheel instance id's

```

wl_data2$wheel_instance_id1 = paste0(wl_data2$Stock, wl_data2$SetNo, wl_data2$CoachNo,
wl_data2$BogieNo, wl_data2$axle_number, wl_data2$wheel_id)

```

```

for(i in 1:nrow(wl_data2)){
  if(wl_data2[i,'measurement_id'] == 1){
    wheel_instance_id_count = 1
    instance_id = paste0('instance-',wheel_instance_id_count)
    wl_data2[i,'wheel_instance_id'] = instance_id
  }else if(wl_data2[i,'wheel_change_flag'] == 1){
    wheel_instance_id_count = wheel_instance_id_count+1
    instance_id = paste0('instance-',wheel_instance_id_count)
    wl_data2[i,'wheel_instance_id'] = instance_id
  }else{
    wl_data2[i,'wheel_instance_id'] = wl_data2[i-1,'wheel_instance_id']
  }
  print(i)
}

```

##drawing samples to check if logic worked

```

# sampleset = wl_data2[wl_data2$measurement_id >= 20,]
# samples = sampleset[sample(1:nrow(sampleset),3),'wheel_instance_id1']

```

```

check = wl_data2[wl_data2$wheel_instance_id1 == '5M2A1113135123',]
check[,c('wheel_instance_id','FH_weekly_change','FH_change','FH','FH_fixed')]
check_1 = check[,c('Datetime','FH','wheel_instance_id')]
check_1$`Measurement Type` = 'Original'
check_2 = check[,c('Datetime','FH_fixed','wheel_instance_id')]
check_2$`Measurement Type` = 'Fixed'
colnames(check_2) = colnames(check_1)
check = rbind(check_1,check_2)
colnames(check)[3] = 'Wheel Instance'

```

```

check2 = wl_data2[wl_data2$wheel_instance_id1 == '5M2A5417625248',]
check2[,c('wheel_instance_id','FH_weekly_change','FH_change','FH','FH_fixed')]
check2_1 = check2[,c('Datetime','FH','wheel_instance_id')]

```



```

check2_1$`Measurement Type` = 'Original'
check2_2 = check2[,c('Datetime','FH_fixed','wheel_instance_id')]
check2_2$`Measurement Type` = 'Fixed'
colnames(check2_2) = colnames(check2_1)
check2 = rbind(check2_1,check2_2)
colnames(check2)[3] = 'Wheel Instance'

```

```

check3 = wl_data2[wl_data2$wheel_instance_id1 == '10M5TC10M50406T124',]
check3[,c('wheel_instance_id','FH_weekly_change','FH_change','FH','FH_fixed')]
check3_1 = check3[,c('Datetime','FH','wheel_instance_id')]
check3_1$`Measurement Type` = 'Original'
check3_2 = check3[,c('Datetime','FH_fixed','wheel_instance_id')]
check3_2$`Measurement Type` = 'Fixed'
colnames(check3_2) = colnames(check3_1)
check3 = rbind(check3_1,check3_2)
colnames(check3)[3] = 'Wheel Instance'

```

```

check4 = wl_data2[wl_data2$wheel_instance_id1 == '10M54910M51028M111',]
check4[,c('wheel_instance_id','FH_weekly_change','FH_change','FH','FH_fixed')]
check4_1 = check4[,c('Datetime','FH','wheel_instance_id')]
check4_1$`Measurement Type` = 'Original'
check4_2 = check4[,c('Datetime','FH_fixed','wheel_instance_id')]
check4_2$`Measurement Type` = 'Fixed'
colnames(check4_2) = colnames(check4_1)
check4 = rbind(check4_1,check4_2)
colnames(check4)[3] = 'Wheel Instance'

```

```

check$Datetime = as.Date(check$Datetime)
check2$Datetime = as.Date(check2$Datetime)
check3$Datetime = as.Date(check3$Datetime)
check4$Datetime = as.Date(check4$Datetime)

```

```

plot4 = ggplot(data = check4, aes(x = Datetime, y = FH, group=`Measurement Type`, colour =
`Measurement Type`)) +
  geom_line() +
  ylim(10, 40) +
  scale_x_date(date_labels = '%Y-%m', date_breaks = '2 month')+
  theme_bw()+
  theme(axis.text.x=element_text(angle=90,vjust=0.5), legend.position="bottom")+
  xlab('Date (yyyy-mm)')+
  ylab('Flange Height (mm)')
plot4

```

```

plot4 = ggplot(data = check4[check4$`Measurement Type` == 'Fixed',], aes(x = Datetime, y =
FH, group=`Wheel Instance`, colour = `Wheel Instance`)) +
  geom_line() +
  ylim(10, 40) +
  scale_x_date(date_labels = '%Y-%m', date_breaks = '2 month')+

```

```

theme_bw()+
theme(axis.text.x=element_text(angle=90,vjust=0.5), legend.position="bottom")+
xlab('Date (yyyy-mm)')+
ylab('Flange Height (mm)')
plot4
# , legend.position="top"

#creating wheel instance counts field
wl_data2$wheel_instance_count = 1
for(i in 2:nrow(wl_data2)){
  if(wl_data2[i,'wheel_instance_id'] == wl_data2[i-1,'wheel_instance_id']){
    wl_data2[i,'wheel_instance_count'] = wl_data2[i-1,'wheel_instance_count']+1
  }else if(wl_data2[i,'wheel_instance_id'] != wl_data2[i-1,'wheel_instance_id']){
    wl_data2[i,'wheel_instance_count'] = 1
  }
  print(i)
}

#dropping wheel instances with less than 3 observations
wl_data2$wheel_instance_count_dropflag = 0
for(i in 1:(nrow(wl_data2)-1)){
  if(wl_data2[i+1,'wheel_instance_count'] == 1){
    if(wl_data2[i,'wheel_instance_count'] < 4){
      count_ = wl_data2[i,'wheel_instance_count']
      wl_data2[(i-(count_-1)):i,'wheel_instance_count_dropflag'] = 1
    }
  }
  print(i)
}

wl_data3 = wl_data2[wl_data2$wheel_instance_count_dropflag == 0,]

#creating engineered features
wl_data3$ID = seq(1:nrow(wl_data3))
drop_fields = colnames(wl_data3)[c(39:49,51,52,56,57)]
wl_data3 = wl_data3[,-c(39:49,51,52,56,57)]

wl_data3$Datetime_change = NA
wl_data3$TD_change = NA
wl_data3$HW_change = NA
wl_data3$FT_change = NA
wl_data3$FS_change = NA
wl_data3$FH_change = NA

##consecutive wheel instance datetime, td, hw, ft, fs and fh change
for(i in 2:nrow(wl_data3)){
  if(wl_data3[i,'wheel_instance_id'] == wl_data3[i-1,'wheel_instance_id']){

```

```

    wl_data3[i,'Datetime_change'] = difftime(wl_data3[i,'Datetime'],wl_data3[i-1,'Datetime'], units =
'days')
    wl_data3[i,'TD_change'] = wl_data3[i,'TD'] - wl_data3[i-1,'TD']
    wl_data3[i,'HW_change'] = wl_data3[i,'HW'] - wl_data3[i-1,'HW']
    wl_data3[i,'FT_change'] = wl_data3[i,'FT'] - wl_data3[i-1,'FT']
    wl_data3[i,'FS_change'] = wl_data3[i,'FS'] - wl_data3[i-1,'FS']
    wl_data3[i,'FH_change'] = wl_data3[i,'FH_fixed'] - wl_data3[i-1,'FH_fixed']
}
print(i)
}

```

##consecutive wheel instance datetime, td, hw, ft, fs and fh change

```

for(i in 2:nrow(wl_data3)){
  if(wl_data3[i,'wheel_instance_id'] == wl_data3[i-1,'wheel_instance_id']){
    wl_data3[i,'Datetime_change'] = difftime(wl_data3[i,'Datetime'],wl_data3[i-1,'Datetime'], units =
'days')
    wl_data3[i,'TD_change'] = wl_data3[i,'TD'] - wl_data3[i-1,'TD']
    wl_data3[i,'HW_change'] = wl_data3[i,'HW'] - wl_data3[i-1,'HW']
    wl_data3[i,'FT_change'] = wl_data3[i,'FT'] - wl_data3[i-1,'FT']
    wl_data3[i,'FS_change'] = wl_data3[i,'FS'] - wl_data3[i-1,'FS']
    wl_data3[i,'FH_change'] = wl_data3[i,'FH_fixed'] - wl_data3[i-1,'FH_fixed']
  }
  print(i)
}

```

```

wl_data3$Datetime_date = as.Date(wl_data3$Datetime)

```

##minimum FH per bogie per measurement

```

min_fh = sqldf('select min(FH_fixed) as min_fh_bogie, Stock, SetNo, CoachNo, BogieNo,
Datetime_date
    from wl_data3
    group by Stock, SetNo, CoachNo, BogieNo,Datetime_date')

```

```

ave_fh = sqldf('select a.*, b.min_avg_fh_bogie
from
(select avg(FH_fixed) as avg_fh_bogie, Stock, SetNo, CoachNo, BogieNo, Datetime_date
    from wl_data3
    group by Stock, SetNo, CoachNo,BogieNo,Datetime_date)a
inner join
(select min(b1.avg_fh_bogie) as min_avg_fh_bogie, b1.Stock as Stock, b1.SetNo as SetNo,
b1.CoachNo as CoachNo, b1.Datetime_date as Datetime_date
from
(select avg(FH_fixed) as avg_fh_bogie, Stock, SetNo, CoachNo,BogieNo, Datetime_date
    from wl_data3
    group by Stock, SetNo, CoachNo, BogieNo, Datetime_date)b1
group by b1.Stock, b1.SetNo, b1.CoachNo,b1.Datetime_date)b
on a.Stock = b.Stock
and a.SetNo = b.SetNo

```

```

and a.CoachNo = b.CoachNo
and a.Datetime_date = b.Datetime_date ')

```

```

ave_fh$avg_fh_bogie_diff = ave_fh$avg_fh_bogie - ave_fh$min_avg_fh_bogie

```

```

min_fh_axle = sqldf('select min(FH_fixed) as min_fh_axle, Stock, SetNo, CoachNo, BogieNo,
axle_number,Datetime_date
from wl_data3
group by Stock, SetNo, CoachNo, BogieNo, axle_number,Datetime_date')

```

```

wl_data3 = sqldf("select
a.*,
b.min_fh_bogie
from wl_data3 a
inner join
min_fh b
on a.Stock = b.Stock
and a.SetNo = b.SetNo
and a.CoachNo = b.CoachNo
and a.BogieNo = b.BogieNo
and a.Datetime_date = b.Datetime_date")

```

```

wl_data3 = sqldf("select a.*, c.min_fh_axle
from wl_data3 a
inner join
min_fh_axle c
on a.Stock = c.Stock
and a.SetNo = c.SetNo
and a.CoachNo = c.CoachNo
and a.BogieNo = c.BogieNo
and a.axle_number = c.axle_number
and a.Datetime_date = c.Datetime_date")

```

```

wl_data3 = sqldf("select a.*, d.avg_fh_bogie_diff
from wl_data3 a
inner join
ave_fh d
on a.Stock = d.Stock
and a.SetNo = d.SetNo
and a.CoachNo = d.CoachNo
and a.BogieNo = d.BogieNo
and a.Datetime_date = d.Datetime_date")

```

```

wl_data3$previous_FH = NA
wl_data3$previous_TD = NA
wl_data3$previous_HW = NA
wl_data3$previous_FT = NA

```

```
wl_data3$previous_FS = NA
```

```
for(i in 2:nrow(wl_data3)){
  if(wl_data3[i,'wheel_instance_id'] == wl_data3[i-1,'wheel_instance_id']){
    wl_data3[i,'previous_FH'] = wl_data3[i-1,'FH_fixed']
    wl_data3[i,'previous_TD'] = wl_data3[i-1,'TD']
    wl_data3[i,'previous_HW'] = wl_data3[i-1,'HW']
    wl_data3[i,'previous_FT'] = wl_data3[i-1,'FT']
    wl_data3[i,'previous_FS'] = wl_data3[i-1,'FS']
  }
  print(i)
}
```

```
wl_data3 = sqldf("select distinct * from wl_data3")
```

```
wl_data3$min_fh_axle_diff = wl_data3$FH_fixed - wl_data3$min_fh_axle
```

```
wl_data3$min_fh_bogie_diff = wl_data3$FH_fixed - wl_data3$min_fh_bogie
```

```
wl_data3$wheel_instance_age = 0
#creating wheel instance counts field
wl_data3$wheel_instance_count = 1
for(i in 2:nrow(wl_data3)){
  if(wl_data3[i,'wheel_instance_id'] == wl_data3[i-1,'wheel_instance_id']){
    wl_data3[i,'wheel_instance_count'] = wl_data3[i-1,'wheel_instance_count']+1
  }else if(wl_data3[i,'wheel_instance_id'] != wl_data3[i-1,'wheel_instance_id']){
    wl_data3[i,'wheel_instance_count'] = 1
  }
  print(i)
}
```

```
#creating wheel instance age
for(i in 1:(nrow(wl_data3))){
  if(wl_data3[i,'wheel_instance_count'] == 1){
    timestamp_ = wl_data3[i,'Datetime']
  }
  wl_data3[i,'wheel_instance_age'] = difftime(timestamp_,wl_data3[i,'Datetime'], units = 'days')
  print(i)
}
```

```
#creating moving average wear variables
wl_data3$FH_moving_average = 0
wl_data3$TD_moving_average = 0
wl_data3$HW_moving_average = 0
wl_data3$FT_moving_average = 0
wl_data3$FS_moving_average = 0
```

```

wl_data3[is.na(wl_data3$FH_change),'FH_change'] = 0
wl_data3[is.na(wl_data3$TD_change),'TD_change'] = 0
wl_data3[is.na(wl_data3$HW_change),'HW_change'] = 0
wl_data3[is.na(wl_data3$FT_change),'FT_change'] = 0
wl_data3[is.na(wl_data3$FS_change),'FS_change'] = 0

for(i in 1:nrow(wl_data3)){
  if(wl_data3[i,'wheel_instance_count'] == 1){
    wl_data3[i,'FH_moving_average'] = 0
    wl_data3[i,'TD_moving_average'] = 0
    wl_data3[i,'HW_moving_average'] = 0
    wl_data3[i,'FT_moving_average'] = 0
    wl_data3[i,'FS_moving_average'] = 0
  }else if(wl_data3[i,'wheel_instance_count'] == 2){
    wl_data3[i,'FH_moving_average'] = wl_data3[i,'FH_change']
    wl_data3[i,'TD_moving_average'] = wl_data3[i,'TD_change']
    wl_data3[i,'HW_moving_average'] = wl_data3[i,'HW_change']
    wl_data3[i,'FT_moving_average'] = wl_data3[i,'FT_change']
    wl_data3[i,'FS_moving_average'] = wl_data3[i,'FH_change']
  }else {
    wl_data3[i,'FH_moving_average'] = mean(wl_data3[i,'FH_change'],wl_data3[i-
1,'FH_change'],wl_data3[i-2,'FH_change'],na.rm = T)
    wl_data3[i,'TD_moving_average'] = mean(wl_data3[i,'TD_change'],wl_data3[i-
1,'TD_change'],wl_data3[i-2,'TD_change'],na.rm = T)
    wl_data3[i,'HW_moving_average'] = mean(wl_data3[i,'HW_change'],wl_data3[i-
1,'HW_change'],wl_data3[i-2,'HW_change'],na.rm = T)
    wl_data3[i,'FT_moving_average'] = mean(wl_data3[i,'FT_change'],wl_data3[i-
1,'FT_change'],wl_data3[i-2,'FT_change'],na.rm = T)
    wl_data3[i,'FS_moving_average'] = mean(wl_data3[i,'FS_change'],wl_data3[i-
1,'FS_change'],wl_data3[i-2,'FS_change'],na.rm = T)
  }
  print(i)
}

#creating target variables
wl_data3$FH_target = 0
wl_data3$TD_target = 0
wl_data3$HW_target = 0
wl_data3$FT_target = 0
wl_data3$FS_target = 0

wl_data3[wl_data3$FH >= 35 & !is.na(wl_data3$FH),'FH_target'] = 1
wl_data3[wl_data3$TD <= 978 & !is.na(wl_data3$TD),'TD_target'] = 1
wl_data3[wl_data3$HW <= -3 & !is.na(wl_data3$HW),'HW_target'] = 1
wl_data3[wl_data3$FT <= 19 & !is.na(wl_data3$FT),'FT_target'] = 1
wl_data3[wl_data3$FS >= 6.5 & !is.na(wl_data3$FS),'FS_target'] = 1

```

```

for(i in 1:(nrow(wl_data3)-1)){
  if(wl_data3[i+1,'wheel_instance_count'] == 1){
    wl_data3[i,'FH_target'] = 1
    wl_data3[i,'TD_target'] = 1
    wl_data3[i,'HW_target'] = 1
    wl_data3[i,'FT_target'] = 1
    wl_data3[i,'FS_target'] = 1
    print(i)
  }
}

#normalising numeric variables
wl_data4 = wl_data3

wl_data4$Datetime_change = (wl_data4$Datetime_change - mean(wl_data4$Datetime_change,
na.rm = T))/sd(wl_data4$Datetime_change, na.rm = T)

wl_data4$TD_change = (wl_data4$TD_change - mean(wl_data4$TD_change, na.rm =
T))/sd(wl_data4$TD_change, na.rm = T)
wl_data4$HW_change = (wl_data4$HW_change - mean(wl_data4$HW_change, na.rm =
T))/sd(wl_data4$HW_change, na.rm = T)
wl_data4$FT_change = (wl_data4$FT_change - mean(wl_data4$FT_change, na.rm =
T))/sd(wl_data4$FT_change, na.rm = T)
wl_data4$FS_change = (wl_data4$FS_change - mean(wl_data4$FS_change, na.rm =
T))/sd(wl_data4$FS_change, na.rm = T)
wl_data4$FH_change = (wl_data4$FH_change - mean(wl_data4$FH_change, na.rm =
T))/sd(wl_data4$FH_change, na.rm = T)

wl_data4$min_fh_bogie = (wl_data4$min_fh_bogie - mean(wl_data4$min_fh_bogie, na.rm =
T))/sd(wl_data4$min_fh_bogie, na.rm = T)
wl_data4$min_fh_axle = (wl_data4$min_fh_axle - mean(wl_data4$min_fh_axle, na.rm =
T))/sd(wl_data4$min_fh_axle, na.rm = T)
wl_data4$avg_fh_bogie_diff = (wl_data4$avg_fh_bogie_diff -
mean(wl_data4$avg_fh_bogie_diff, na.rm = T))/sd(wl_data4$avg_fh_bogie_diff, na.rm = T)
wl_data4$min_fh_axle_diff = (wl_data4$min_fh_axle_diff - mean(wl_data4$min_fh_axle_diff,
na.rm = T))/sd(wl_data4$min_fh_axle_diff, na.rm = T)
wl_data4$min_fh_bogie_diff = (wl_data4$min_fh_bogie_diff -
mean(wl_data4$min_fh_bogie_diff, na.rm = T))/sd(wl_data4$min_fh_bogie_diff, na.rm = T)

wl_data4$previous_FH = (wl_data4$previous_FH - mean(wl_data4$previous_FH, na.rm =
T))/sd(wl_data4$previous_FH, na.rm = T)
wl_data4$previous_TD = (wl_data4$previous_TD - mean(wl_data4$previous_TD, na.rm =
T))/sd(wl_data4$previous_TD, na.rm = T)
wl_data4$previous_HW = (wl_data4$previous_HW - mean(wl_data4$previous_HW, na.rm =
T))/sd(wl_data4$previous_HW, na.rm = T)
wl_data4$previous_FT = (wl_data4$previous_FT - mean(wl_data4$previous_FT, na.rm =
T))/sd(wl_data4$previous_FT, na.rm = T)

```

```
wl_data4$previous_FS = (wl_data4$previous_FS - mean(wl_data4$previous_FS, na.rm =
T))/sd(wl_data4$previous_FS, na.rm = T)
```

```
wl_data4$wheel_instance_age = (wl_data4$wheel_instance_age -
mean(wl_data4$wheel_instance_age, na.rm = T))/sd(wl_data4$wheel_instance_age, na.rm = T)
```

```
wl_data4$FH_moving_average = (wl_data4$FH_moving_average -
mean(wl_data4$FH_moving_average, na.rm = T))/sd(wl_data4$FH_moving_average, na.rm =
T)
```

```
wl_data4$TD_moving_average = (wl_data4$TD_moving_average -
mean(wl_data4$TD_moving_average, na.rm = T))/sd(wl_data4$TD_moving_average, na.rm =
T)
```

```
wl_data4$HW_moving_average = (wl_data4$HW_moving_average -
mean(wl_data4$HW_moving_average, na.rm = T))/sd(wl_data4$HW_moving_average, na.rm =
T)
```

```
wl_data4$FT_moving_average = (wl_data4$FT_moving_average -
mean(wl_data4$FT_moving_average, na.rm = T))/sd(wl_data4$FT_moving_average, na.rm =
T)
```

```
wl_data4$FS_moving_average = (wl_data4$FS_moving_average -
mean(wl_data4$FS_moving_average, na.rm = T))/sd(wl_data4$FS_moving_average, na.rm =
T)
```

```
#creating passing months variables
```

```
monnb <- function(d) { lt <- as.POSIXlt(as.Date(d, origin="1900-01-01"));
lt$year*12 + lt$mon }
```

```
mondf <- function(d1, d2) { monnb(d2) - monnb(d1) }
```

```
month_pass = data.frame(matrix(nrow = nrow(wl_data4), ncol = 12, data = 0))
```

```
colnames(month_pass) = c('Jan','Feb','Mar','Apr','May','Jun','Jul','Aug','Sep','Oct','Nov','Dec')
```

```
for(i in 107562:nrow(wl_data4)){
```

```
  if(wl_data4[i,'wheel_instance_id'] == wl_data4[i-1,'wheel_instance_id']){
```

```
    if(wl_data4[i,'measurement_id'] != 1){
```

```
      month_diff = mondf(wl_data4[i-1,'Datetime_date'], wl_data4[i,'Datetime_date'])
```

```
      curmonth_index = grep(months(wl_data4[i,'Datetime_date'],abbreviate = T)[1],
```

```
colnames(month_pass))
```

```
      if((month_diff+curmonth_index) <= 12){
```

```
        month_pass[i,c(curmonth_index:(month_diff+curmonth_index))] = 1
```

```
      }else if((month_diff+curmonth_index) > 12 & (month_diff+curmonth_index) < 24){
```

```
        month_diff1 = month_diff+curmonth_index-12
```

```
        month_pass[i,c(curmonth_index:ncol(month_pass))] = 1
```

```
        month_pass[i,c(1:month_diff1)] = 1
```

```
      } else if((month_diff+curmonth_index) > 24){
```

```
        month_pass[i,c(1:ncol(month_pass))] = 1
```

```
      }
```

```
    }
```

```
  }
```

```
print(i)
```



```

}

colnames(month_pass) =
c('Jan_passed','Feb_passed','Mar_passed','Apr_passed','May_passed','Jun_passed','Jul_passe
d','Aug_passed','Sep_passed','Oct_passed','Nov_passed','Dec_passed')
wl_data4 = cbind(wl_data4, month_pass)

### Modelling
#creating train and test datasets
train_index = sample(1:nrow(wl_data4), 0.7*nrow(wl_data4))
ml_cols = c(14:37,43:48,50:60,62:83)
train = wl_data4[train_index,ml_cols]
test = wl_data4[-c(train_index),ml_cols]

#model building
##logistic regression
library(caret)
library(ROCR)

#modelling FH

train_FH = train[,-c(48:51)]
test_FH = test[,-c(48:51)]
logistic_model_FH = glm(train_FH$FH_target~.,family = binomial(link = "logit"),data = train_FH)

#testing FH model
prediction = predict(logistic_model_FH,newdata = test_FH, type = 'response')
fitted_logisitic_FH = data.frame(prediction)
colnames(fitted_logisitic_FH) = 'Predicted'
fitted_logisitic_FH$Predicted = ifelse(fitted_logisitic_FH$Predicted > 0.5,1,0)

fitted_logisitic_FH$Predicted = as.factor(fitted_logisitic_FH$Predicted)
test_FH$FH_target = as.factor(test_FH$FH_target)

#evaluating test FH model
##CM
conf_mat_FH = caret::confusionMatrix(fitted_logisitic_FH$Predicted, test_FH$FH_target)
conf_mat_FH
##ROC
pr = ROCR::prediction(prediction, test_FH$FH_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)

#modelling TD
train_TD = train[,-c(47,49:51)]
test_TD = test[,-c(47,49:51)]
logistic_model_TD = glm(train_TD$TD_target~.,family = binomial(link = "logit"),data = train_TD)

```

```

#testing TD model
prediction = predict(logistic_model_TD,newdata = test_TD, type = 'response')
fitted_logisitic_TD = data.frame(prediction)
colnames(fitted_logisitic_TD) = 'Predicted'
fitted_logisitic_TD$Predicted = ifelse(fitted_logisitic_TD$Predicted > 0.5,1,0)

fitted_logisitic_TD$Predicted = as.factor(fitted_logisitic_TD$Predicted)
test_TD$TD_target = as.factor(test_TD$TD_target)

#evaluating test TD model
##CM
conf_mat_TD = caret::confusionMatrix(fitted_logisitic_TD$Predicted, test_TD$TD_target)
conf_mat_TD
##ROC
pr = ROCR::prediction(prediction, test_TD$TD_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)

#modelling HW
train_HW = train[,-c(47,48,50:51)]
test_HW = test[,-c(47,48,50:51)]
logistic_model_HW = glm(train_HW$HW_target~.,family = binomial(link = "logit"),data =
train_HW)

#testing HW model
prediction = predict(logistic_model_HW,newdata = test_HW, type = 'response')
fitted_logisitic_HW = data.frame(prediction)
colnames(fitted_logisitic_HW) = 'Predicted'
fitted_logisitic_HW$Predicted = ifelse(fitted_logisitic_HW$Predicted > 0.5,1,0)

fitted_logisitic_HW$Predicted = as.factor(fitted_logisitic_HW$Predicted)
test_HW$HW_target = as.factor(test_HW$HW_target)

#evaluating test HW model
##CM
conf_mat_HW = caret::confusionMatrix(fitted_logisitic_HW$Predicted, test_HW$HW_target)
conf_mat_HW
##ROC
pr = ROCR::prediction(prediction, test_HW$HW_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)

#modelling FT
train_FT = train[,-c(47,48,49,51)]
test_FT = test[,-c(47,48,49,51)]
logistic_model_FT = glm(train_FT$FT_target~.,family = binomial(link = "logit"),data = train_FT)

#testing FT model

```

```

prediction = predict(logistic_model_FT,newdata = test_FT, type = 'response')
fitted_logisitic_FT = data.frame(prediction)
colnames(fitted_logisitic_FT) = 'Predicted'
fitted_logisitic_FT$Predicted = ifelse(fitted_logisitic_FT$Predicted > 0.5,1,0)

fitted_logisitic_FT$Predicted = as.factor(fitted_logisitic_FT$Predicted)
test_FT$FT_target = as.factor(test_FT$FT_target)

#evaluating test FT model
##CM
conf_mat_FT = caret::confusionMatrix(fitted_logisitic_FT$Predicted, test_FT$FT_target)
conf_mat_FT
##ROC
pr = ROCR::prediction(prediction, test_FT$FT_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)

#modelling FS
train_FS = train[,-c(47:50)]
test_FS = test[,-c(47:50)]
logistic_model_FS = glm(train_FS$FS_target~.,family = binomial(link = "logit"),data = train_FS)

#testing FS model
prediction = predict(logistic_model_FS,newdata = test_FS, type = 'response')
fitted_logisitic_FS = data.frame(prediction)
colnames(fitted_logisitic_FS) = 'Predicted'
fitted_logisitic_FS$Predicted = ifelse(fitted_logisitic_FS$Predicted > 0.5,1,0)

fitted_logisitic_FS$Predicted = as.factor(fitted_logisitic_FS$Predicted)
test_FS$FS_target = as.factor(test_FS$FS_target)

#evaluating test FS model
##CM
conf_mat_FS = caret::confusionMatrix(fitted_logisitic_FS$Predicted, test_FS$FS_target)
conf_mat_FS
##ROC
pr = ROCR::prediction(prediction, test_FS$FS_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)

##ANN
library(ANN2)

momentum = data.frame('momentum' = c(0.1,0.3,0.5))
learning_rate = data.frame('learning_rate' = c(0.005,0.05,0.5))
first_hidden_layers = data.frame('first_hidden' = seq(3,10,1))
second_hidden_layers = data.frame('second_hidden' = c(0,seq(3,10,1)))

```

```

#ANN FH
iter = 1
max_iter =
nrow(momentum)*nrow(learning_rate)*nrow(first_hidden_layers)*nrow(second_hidden_layers)

train_FH = train[,-c(48:51)]
test_FH = test[,-c(48:51)]

train_FH[,1:ncol(train_FH)] = lapply(train_FH[,1:ncol(train_FH)], as.numeric)
test_FH[,1:ncol(test_FH)] = lapply(test_FH[,1:ncol(test_FH)], as.numeric)

#nn
nn_FH_performance = data.frame(t(conf_mat_FH$byClass))
avg_nn_FH_performance_overall = nn_FH_performance
avg_nn_FH_performance_overall$ID = 0
avg_nn_FH_performance_overall = avg_nn_FH_performance_overall[0,]
nn_FH_performance = nn_FH_performance[0,]
avg_nn_FH_performance = nn_FH_performance

# k_folds_set_index = sample(1:nrow(train_FH), 0.5*nrow(train_FH))

train_FH_KF = train[k_folds_set_index,-c(48:51)]
train_FH_KF[,c(1:24)] = lapply(train_FH_KF[,1:24], as.character)
train_FH_KF[,1:ncol(train_FH_KF)] = lapply(train_FH_KF[,1:ncol(train_FH_KF)], as.numeric)

for(m in 1:nrow(momentum)){
  for(l in 1:nrow(learning_rate)){
    for(fh in 1:nrow(first_hidden_layers)){
      for(sh in 1:nrow(second_hidden_layers)){
        print('% finished:')
        print(iter/max_iter*100)
        nn_FH_performance = nnconf_mat_FH[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FH_KF),round(nrow(train_FH_KF)*1/3), replace = F)
          holdout = train_FH_KF[k_index,]
          remainder = train_FH_KF[-k_index,]
          if(second_hidden_layers[sh,1] == 0){
            nn_FH <- neuralnetwork(remainder[,-c(which(colnames(remainder) ==
'FH_target'))],remainder[,c(which(colnames(remainder) == 'FH_target'))], hiddenLayers =
c(first_hidden_layers[fh,1]), maxEpochs = 20, momentum = momentum[m,1],learnRate =
learning_rate[l,1],verbose = F,sigmoidLayers = c(1), lossFunction = "log")
          }else{
            nn_FH <- neuralnetwork(remainder[,-c(which(colnames(remainder) ==
'FH_target'))],remainder[,c(which(colnames(remainder) == 'FH_target'))], hiddenLayers =
c(first_hidden_layers[fh,1],second_hidden_layers[sh,1]), maxEpochs = 20, momentum =

```

```

momentum[m,1],learnRate = learning_rate[l,1],verbose = F,sigmoidLayers = c(1,2), lossFunction
= "log")
  }
  prediction = as.data.frame(predict(nn_FH,holdout[, -c(which(colnames(holdout) ==
'FH_target'))]))[1]
  prediction$predictions = as.numeric(as.character(prediction$predictions))
  prediction$predictions = as.factor(prediction$predictions)
  holdout$FH_target = as.factor(holdout$FH_target)
  nnconf_mat_FH = caret::confusionMatrix(prediction$predictions, holdout$FH_target)
  nnconf_mat_FH = as.data.frame(t(nnconf_mat_FH$byClass))
  nn_FH_performance = rbind(nn_FH_performance, nnconf_mat_FH)
}
avg_nn_FH_performance = data.frame(t(colMeans(nn_FH_performance)))
avg_nn_FH_performance$ID = iter
avg_nn_FH_performance_overall = rbind(avg_nn_FH_performance_overall,
avg_nn_FH_performance)
iter = iter+1
}
}
}
}

```

```

#save.image(file='C:/Users/fkimokoti.000/Desktop/PRASA
DATA/FRED/johan_workspace.RData')

```

```

library(ggplot2)
plot = ggplot(avg_nn_FH_performance_overall, aes(x = ID, y = F1)) +
  geom_line() +
  ylab('Average F1 Score') +
  xlab('Hyperparameter Configuration ID')
plot

```

```

#ANN TD
iter = 1
max_iter =
nrow(momentum)*nrow(learning_rate)*nrow(first_hidden_layers)*nrow(second_hidden_layers)

```

```

train_TD = train[, -c(47,49:51)]
test_TD = test[, -c(47,49:51)]

```

```

train_TD[, 1:ncol(train_TD)] = lapply(train_TD[, 1:ncol(train_TD)], as.numeric)
test_TD[, 1:ncol(test_TD)] = lapply(test_TD[, 1:ncol(test_TD)], as.numeric)

```

```

#nn
nn_TD_performance = data.frame(t(conf_mat_TD$byClass))
avg_nn_TD_performance_overall = nn_TD_performance

```

```

avg_nn_TD_performance_overall$ID = 0
avg_nn_TD_performance_overall = avg_nn_TD_performance_overall[0,]
nn_TD_performance = nn_TD_performance[0,]
avg_nn_TD_performance = nn_TD_performance
nnconf_mat_TD = nnconf_mat_FH[0,]
# k_folds_set_index = sample(1:nrow(train_TD), 0.5*nrow(train_TD))

train_TD_KF = train[k_folds_set_index,-c(47,49:51)]
train_TD_KF[,c(1:24)] = lapply(train_TD_KF[,1:24], as.character)
train_TD_KF[,1:ncol(train_TD_KF)] = lapply(train_TD_KF[,1:ncol(train_TD_KF)], as.numeric)

for(m in 1:nrow(momentum)){
  for(l in 1:nrow(learning_rate)){
    for(TD in 1:nrow(first_hidden_layers)){
      for(sh in 1:nrow(second_hidden_layers)){
        print('% finished:')
        print(iter/max_iter*100)
        nn_TD_performance = nnconf_mat_TD[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_TD_KF),round(nrow(train_TD_KF)*1/3), replace = F)
          holdout = train_TD_KF[k_index,]
          remainder = train_TD_KF[-k_index,]
          if(second_hidden_layers[sh,1] == 0){
            nn_TD <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'TD_target'))],remainder[,c(which(colnames(remainder) == 'TD_target'))], hiddenLayers =
c(first_hidden_layers[TD,1]), maxEpochs = 20, momentum = momentum[m,1],learnRate =
learning_rate[l,1],verbose = F,sigmoidLayers = c(1), lossFunction = "log")
          }else{
            nn_TD <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'TD_target'))],remainder[,c(which(colnames(remainder) == 'TD_target'))], hiddenLayers =
c(first_hidden_layers[TD,1],second_hidden_layers[sh,1]), maxEpochs = 20, momentum =
momentum[m,1],learnRate = learning_rate[l,1],verbose = F,sigmoidLayers = c(1,2), lossFunction
= "log")
          }
          prediction = as.data.frame(predict(nn_TD,holdout[, -c(which(colnames(holdout) ==
'TD_target'))]))[1]
          prediction$predictions = as.numeric(as.character(prediction$predictions))
          prediction$predictions = as.factor(prediction$predictions)
          holdout$TD_target = as.factor(holdout$TD_target)
          nnconf_mat_TD = caret::confusionMatrix(prediction$predictions, holdout$TD_target)
          nnconf_mat_TD = as.data.frame(t(nnconf_mat_TD$byClass))
          nn_TD_performance = rbind(nn_TD_performance,nnconf_mat_TD)
        }
        avg_nn_TD_performance = data.frame(t(colMeans(nn_TD_performance)))
        avg_nn_TD_performance$ID = iter
        avg_nn_TD_performance_overall = rbind(avg_nn_TD_performance_overall,
avg_nn_TD_performance)
      }
    }
  }
}

```

```

        iter = iter+1
      }
    }
  }
}

#ANN HW
iter = 1
max_iter =
nrow(momentum)*nrow(learning_rate)*nrow(first_hidden_layers)*nrow(second_hidden_layers)

train_HW = train[,-c(47,48,50:51)]
test_HW = test[,-c(47,48,50:51)]

train_HW[,1:ncol(train_HW)] = lapply(train_HW[,1:ncol(train_HW)], as.numeric)
test_HW[,1:ncol(test_HW)] = lapply(test_HW[,1:ncol(test_HW)], as.numeric)

#nn
nn_HW_performance = data.frame(t(conf_mat_HW$byClass))
avg_nn_HW_performance_overall = nn_HW_performance
avg_nn_HW_performance_overall$ID = 0
avg_nn_HW_performance_overall = avg_nn_HW_performance_overall[0,]
nn_HW_performance = nn_HW_performance[0,]
avg_nn_HW_performance = nn_HW_performance
nnconf_mat_HW = nnconf_mat_FH[0,]
# k_folds_set_index = sample(1:nrow(train_HW), 0.5*nrow(train_HW))

train_HW_KF = train[k_folds_set_index,-c(47,48,50:51)]
train_HW_KF[,c(1:24)] = lapply(train_HW_KF[,1:24], as.character)
train_HW_KF[,1:ncol(train_HW_KF)] = lapply(train_HW_KF[,1:ncol(train_HW_KF)], as.numeric)

for(m in 1:nrow(momentum)){
  for(l in 1:nrow(learning_rate)){
    for(HW in 1:nrow(first_hidden_layers)){
      for(sh in 1:nrow(second_hidden_layers)){
        print('% finished:')
        print(iter/max_iter*100)
        nn_HW_performance = nnconf_mat_HW[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_HW_KF),round(nrow(train_HW_KF)*1/3), replace = F)
          holdout = train_HW_KF[k_index,]
          remainder = train_HW_KF[-k_index,]
          if(second_hidden_layers[sh,1] == 0){
            nn_HW <- neuralnetwork(remainder[,c(which(colnames(remainder) ==
'HW_target'))],remainder[,c(which(colnames(remainder) == 'HW_target'))], hiddenLayers =

```

```

c(first_hidden_layers[HW,1]), maxEpochs = 20, momentum = momentum[m,1],learnRate =
learning_rate[l,1],verbose = F,sigmoidLayers = c(1), lossFunction = "log")
  }else{
    nn_HW <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'HW_target'))], remainder[, c(which(colnames(remainder) == 'HW_target'))], hiddenLayers =
c(first_hidden_layers[HW,1], second_hidden_layers[sh,1]), maxEpochs = 20, momentum =
momentum[m,1], learnRate = learning_rate[l,1], verbose = F, sigmoidLayers = c(1,2), lossFunction
= "log")
  }
  prediction = as.data.frame(predict(nn_HW, holdout[, -c(which(colnames(holdout) ==
'HW_target'))]))[1]
  prediction$predictions = as.numeric(as.character(prediction$predictions))
  prediction$predictions = as.factor(prediction$predictions)
  holdout$HW_target = as.factor(holdout$HW_target)
  nnconf_mat_HW = caret::confusionMatrix(prediction$predictions, holdout$HW_target)
  nnconf_mat_HW = as.data.frame(t(nnconf_mat_HW$byClass))
  nn_HW_performance = rbind(nn_HW_performance, nnconf_mat_HW)
}
avg_nn_HW_performance = data.frame(t(colMeans(nn_HW_performance)))
avg_nn_HW_performance$ID = iter
avg_nn_HW_performance_overall = rbind(avg_nn_HW_performance_overall,
avg_nn_HW_performance)
iter = iter+1
}
}
}
}

```

```
#ANN FS
```

```
iter = 1
```

```
max_iter =
```

```
nrow(momentum)*nrow(learning_rate)*nrow(first_hidden_layers)*nrow(second_hidden_layers)
```

```
train_FS = train[, -c(47:50)]
```

```
test_FS = test[, -c(47:50)]
```

```
train_FS[, 1:ncol(train_FS)] = lapply(train_FS[, 1:ncol(train_FS)], as.numeric)
```

```
test_FS[, 1:ncol(test_FS)] = lapply(test_FS[, 1:ncol(test_FS)], as.numeric)
```

```
#nn
```

```
nn_FS_performance = data.frame(t(conf_mat_FS$byClass))
```

```
avg_nn_FS_performance_overall = nn_FS_performance
```

```
avg_nn_FS_performance_overall$ID = 0
```

```
avg_nn_FS_performance_overall = avg_nn_FS_performance_overall[0,]
```

```
nn_FS_performance = nn_FS_performance[0,]
```

```
avg_nn_FS_performance = nn_FS_performance
```

```
nnconf_mat_FS = nnconf_mat_FH[0,]
```



```

# k_folds_set_index = sample(1:nrow(train_FS), 0.5*nrow(train_FS))

train_FS_KF = train[k_folds_set_index,-c(47:50)]
train_FS_KF[,c(1:24)] = lapply(train_FS_KF[,1:24], as.character)
train_FS_KF[,1:ncol(train_FS_KF)] = lapply(train_FS_KF[,1:ncol(train_FS_KF)], as.numeric)

for(m in 1:nrow(momentum)){
  for(l in 1:nrow(learning_rate)){
    for(FS in 1:nrow(first_hidden_layers)){
      for(sh in 1:nrow(second_hidden_layers)){
        print('% finished:')
        print(iter/max_iter*100)
        nn_FS_performance = nnconf_mat_FS[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FS_KF),round(nrow(train_FS_KF)*1/3), replace = F)
          holdout = train_FS_KF[k_index,]
          remainder = train_FS_KF[-k_index,]
          if(second_hidden_layers[sh,1] == 0){
            nn_FS <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'FS_target'))], remainder[, c(which(colnames(remainder) == 'FS_target'))], hiddenLayers =
c(first_hidden_layers[FS,1]), maxEpochs = 20, momentum = momentum[m,1], learnRate =
learning_rate[l,1], verbose = F, sigmoidLayers = c(1), lossFunction = "log")
          }else{
            nn_FS <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'FS_target'))], remainder[, c(which(colnames(remainder) == 'FS_target'))], hiddenLayers =
c(first_hidden_layers[FS,1], second_hidden_layers[sh,1]), maxEpochs = 20, momentum =
momentum[m,1], learnRate = learning_rate[l,1], verbose = F, sigmoidLayers = c(1,2), lossFunction
= "log")
          }
          prediction = as.data.frame(predict(nn_FS, holdout[, -c(which(colnames(holdout) ==
'FS_target'))]))[1]
          prediction$predictions = as.numeric(as.character(prediction$predictions))
          prediction$predictions = as.factor(prediction$predictions)
          holdout$FS_target = as.factor(holdout$FS_target)
          nnconf_mat_FS = caret::confusionMatrix(prediction$predictions, holdout$FS_target)
          nnconf_mat_FS = as.data.frame(t(nnconf_mat_FS$byClass))
          nn_FS_performance = rbind(nn_FS_performance, nnconf_mat_FS)
        }
        avg_nn_FS_performance = data.frame(t(colMeans(nn_FS_performance)))
        avg_nn_FS_performance$ID = iter
        avg_nn_FS_performance_overall = rbind(avg_nn_FS_performance_overall,
avg_nn_FS_performance)
        iter = iter+1
      }
    }
  }
}

```

```

#ANN FT
iter = 1
max_iter =
nrow(momentum)*nrow(learning_rate)*nrow(first_hidden_layers)*nrow(second_hidden_layers)

train_FT = train[,-c(47,48,49,51)]
test_FT = test[,-c(47,48,49,51)]

train_FT[,1:ncol(train_FT)] = lapply(train_FT[,1:ncol(train_FT)], as.numeric)
test_FT[,1:ncol(test_FT)] = lapply(test_FT[,1:ncol(test_FT)], as.numeric)

#nn
nn_FT_performance = data.frame(t(conf_mat_FT$byClass))
avg_nn_FT_performance_overall = nn_FT_performance
avg_nn_FT_performance_overall$ID = 0
avg_nn_FT_performance_overall = avg_nn_FT_performance_overall[0,]
nn_FT_performance = nn_FT_performance[0,]
avg_nn_FT_performance = nn_FT_performance
nnconf_mat_FT = nnconf_mat_FH[0,]
# k_folds_set_index = sample(1:nrow(train_FT), 0.5*nrow(train_FT))

train_FT_KF = train[k_folds_set_index,-c(47,48,49,51)]
train_FT_KF[,c(1:24)] = lapply(train_FT_KF[,1:24], as.character)
train_FT_KF[,1:ncol(train_FT_KF)] = lapply(train_FT_KF[,1:ncol(train_FT_KF)], as.numeric)

for(m in 1:nrow(momentum)){
  for(l in 1:nrow(learning_rate)){
    for(FT in 1:nrow(first_hidden_layers)){
      for(sh in 1:nrow(second_hidden_layers)){
        print('% finished:')
        print(iter/max_iter*100)
        nn_FT_performance = nnconf_mat_FT[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FT_KF),round(nrow(train_FT_KF)*1/3), replace = F)
          holdout = train_FT_KF[k_index,]
          remainder = train_FT_KF[-k_index,]
          if(second_hidden_layers[sh,1] == 0){
            nn_FT <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'FT_target'))],remainder[,c(which(colnames(remainder) == 'FT_target'))], hiddenLayers =
c(first_hidden_layers[FT,1]), maxEpochs = 20, momentum = momentum[m,1],learnRate =
learning_rate[l,1],verbose = F,sigmoidLayers = c(1), lossFunction = "log")
          }else{
            nn_FT <- neuralnetwork(remainder[, -c(which(colnames(remainder) ==
'FT_target'))],remainder[,c(which(colnames(remainder) == 'FT_target'))], hiddenLayers =
c(first_hidden_layers[FT,1],second_hidden_layers[sh,1]), maxEpochs = 20, momentum =

```

```

momentum[m,1],learnRate = learning_rate[l,1],verbose = F,sigmoidLayers = c(1,2), lossFunction
= "log")
    }
    prediction = as.data.frame(predict(nn_FT,holdout[,-c(which(colnames(holdout) ==
'FT_target'))]))[1]
    prediction$predictions = as.numeric(as.character(prediction$predictions))
    prediction$predictions = as.factor(prediction$predictions)
    holdout$FT_target = as.factor(holdout$FT_target)
    nnconf_mat_FT = caret::confusionMatrix(prediction$predictions, holdout$FT_target)
    nnconf_mat_FT = as.data.frame(t(nnconf_mat_FT$byClass))
    nn_FT_performance = rbind(nn_FT_performance,nnconf_mat_FT)
  }
  avg_nn_FT_performance = data.frame(t(colMeans(nn_FT_performance)))
  avg_nn_FT_performance$ID = iter
  avg_nn_FT_performance_overall = rbind(avg_nn_FT_performance_overall,
avg_nn_FT_performance)
  iter = iter+1
}
}
}
}

```

#ANN FIN

#FH

```
train_FH = train[,-c(48:51)]
```

```
test_FH = test[,-c(48:51)]
```

```
train_FH[,1:ncol(train_FH)] = lapply(train_FH[,1:ncol(train_FH)], as.numeric)
```

```
test_FH[,1:ncol(test_FH)] = lapply(test_FH[,1:ncol(test_FH)], as.numeric)
```

#nn

```
nn_FH_fin <- neuralnetwork(train_FH[,-c(which(colnames(train_FH) ==
'FH_target'))],train_FH[,c(which(colnames(train_FH) == 'FH_target'))], hiddenLayers = c(8,5),
maxEpochs = 1000, momentum = 0.3,learnRate =0.005 ,verbose = T,sigmoidLayers = c(1,2),
lossFunction = "log")
```

```
mm_FH_perf = data.frame(predict(nn_FH_fin,test_FH[,-c(which(colnames(test_FH) ==
'FH_target'))]))
```

##CM

```
conf_mat_FH_nn = caret::confusionMatrix(mm_FH_perf$predictions,
as.factor(test_FH$FH_target))
```

```
conf_mat_FH_nn
```

##ROC

```
pr = ROCR::prediction(as.numeric(as.character(mm_FH_perf$probabilities.class_1)),
test_FH$FH_target)
```

```
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
```

```

plot(prf)

auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

#TD
train_TD = train[,-c(47,49:51)]
test_TD = test[,-c(47,49:51)]

train_TD[,1:ncol(train_TD)] = lapply(train_TD[,1:ncol(train_TD)], as.numeric)
test_TD[,1:ncol(test_TD)] = lapply(test_TD[,1:ncol(test_TD)], as.numeric)

#nn
nn_TD_fin <- neuralnetwork(train_TD[,-c(which(colnames(train_TD) ==
'TD_target'))],train_TD[,c(which(colnames(train_TD) == 'TD_target'))], hiddenLayers = c(3),
maxEpochs = 1000, momentum = 0.3,learnRate =0.05 ,verbose = T,sigmoidLayers = c(1),
lossFunction = "log")
mm_TD_perf = data.frame(predict(nn_TD_fin,test_TD[,-c(which(colnames(test_TD) ==
'TD_target'))]))

##CM

conf_mat_TD_nn = caret::confusionMatrix(mm_TD_perf$predictions,
as.factor(test_TD$TD_target))
conf_mat_TD_nn

##ROC
pr = ROCR::prediction(as.numeric(as.character(mm_TD_perf$probabilities.class_1)),
test_TD$TD_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

#FS
train_FS = train[,-c(47:50)]
test_FS = test[,-c(47:50)]

train_FS[,1:ncol(train_FS)] = lapply(train_FS[,1:ncol(train_FS)], as.numeric)
test_FS[,1:ncol(test_FS)] = lapply(test_FS[,1:ncol(test_FS)], as.numeric)

#nn
nn_FS_fin <- neuralnetwork(train_FS[,-c(which(colnames(train_FS) ==
'FS_target'))],train_FS[,c(which(colnames(train_FS) == 'FS_target'))], hiddenLayers = c(5,5),
maxEpochs = 500, momentum = 0.5,learnRate =0.5 ,verbose = T,sigmoidLayers = c(1,2),
lossFunction = "log")

```

```
mm_FS_perf = data.frame(predict(nn_FS_fin,test_FS[, -c(which(colnames(test_FS) ==
'FS_target'))]))
```

```
##CM
```

```
conf_mat_FS_nn = caret::confusionMatrix(mm_FS_perf$predictions,
as.factor(test_FS$FS_target))
conf_mat_FS_nn
```

```
##ROC
```

```
pr = ROCR::prediction(as.numeric(as.character(mm_FS_perf$probabilities.class_1)),
test_FS$FS_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
#HW
```

```
train_HW = train[, -c(47,48,50:51)]
test_HW = test[, -c(47,48,50:51)]
```

```
train_HW[, 1:ncol(train_HW)] = lapply(train_HW[, 1:ncol(train_HW)], as.numeric)
test_HW[, 1:ncol(test_HW)] = lapply(test_HW[, 1:ncol(test_HW)], as.numeric)
```

```
#nn
```

```
nn_HW_fin <- neuralnetwork(train_HW[, -c(which(colnames(train_HW) ==
'HW_target'))], train_HW[, c(which(colnames(train_HW) == 'HW_target'))], hiddenLayers = c(6,3),
maxEpochs = 500, momentum = 0.005, learnRate = 0.1, verbose = T, sigmoidLayers = c(1,2),
lossFunction = "log")
mm_HW_perf = data.frame(predict(nn_HW_fin, test_HW[, -c(which(colnames(test_HW) ==
'HW_target'))]))
```

```
##CM
```

```
conf_mat_HW_nn = caret::confusionMatrix(mm_HW_perf$predictions,
as.factor(test_HW$HW_target))
conf_mat_HW_nn
```

```
##ROC
```

```
pr = ROCR::prediction(as.numeric(as.character(mm_HW_perf$probabilities.class_1)),
test_HW$HW_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc
```

```
#FT
```

```
train_FT = train[, -c(47,48,49,51)]
```

```

test_FT = test[,-c(47,48,49,51)]

train_FT[,1:ncol(train_FT)] = lapply(train_FT[,1:ncol(train_FT)], as.numeric)
test_FT[,1:ncol(test_FT)] = lapply(test_FT[,1:ncol(test_FT)], as.numeric)

#nn
nn_FT_fin <- neuralnetwork(train_FT[,-c(which(colnames(train_FT) ==
'FT_target'))],train_FT[,c(which(colnames(train_FT) == 'FT_target'))], hiddenLayers = c(6,3),
maxEpochs = 500, momentum = 0.005,learnRate =0.1 ,verbose = T,sigmoidLayers = c(1,2),
lossFunction = "log")
mm_FT_perf = data.frame(predict(nn_FT_fin,test_FT[,-c(which(colnames(test_FT) ==
'FT_target'))]))

##CM

conf_mat_FT_nn = caret::confusionMatrix(mm_FT_perf$predictions,
as.factor(test_FT$FT_target))
conf_mat_FT_nn
##ROC
pr = ROCR::prediction(as.numeric(as.character(mm_FT_perf$probabilities.class_1)),
test_FT$FT_target)
prf = performance(pr, measure = 'tpr', x.measure = 'fpr')
plot(prf)
auc <- performance(pr, measure = "auc")
auc <- auc@y.values[[1]]
auc

##Random Forrest
library(ranger)

bframe = data.frame('b' = c(500,550,600,650,700))
kframe = data.frame('k' = c(round(1/6*ncol(train)),round(1/4*ncol(train)),round(1/3*ncol(train))))
lframe = data.frame('l' = c(1/2,2/3,3/4,4/5))
nframe = data.frame('n' = c(3,5,7,9))

#RF FH
iter = 1
max_iter = nrow(bframe)*nrow(kframe)*nrow(lframe)*nrow(nframe)

train_FH = train[,-c(48:51)]
test_FH = test[,-c(48:51)]

#nn
rf_FH_performance = data.frame(t(conf_mat_FH$byClass))
avg_rf_FH_performance_overall = rf_FH_performance
avg_rf_FH_performance_overall$ID = 0

```

```

avg_rf_FH_performance_overall = avg_rf_FH_performance_overall[0,]
rf_FH_performance = rf_FH_performance[0,]
avg_rf_FH_performance = rf_FH_performance

# k_folds_set_index = sample(1:nrow(train_FH), 0.5*nrow(train_FH))

train_FH_KF = train[k_folds_set_index,-c(48:51)]
train_FH_KF[,c(1:24)] = lapply(train_FH_KF[,1:24], as.character)
train_FH_KF[,1:ncol(train_FH_KF)] = lapply(train_FH_KF[,1:ncol(train_FH_KF)], as.numeric)

for(b in 1:nrow(bframe)){
  for(k in 1:nrow(kframe)){
    for(l in 1:nrow(lframe)){
      for(n in 1:nrow(nframe)){
        print('% finished:')
        print(iter/max_iter*100)
        rf_FH_performance = rf_FH_performance[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FH_KF),round(nrow(train_FH_KF)*1/3), replace = F)
          holdout = train_FH_KF[k_index,]
          remainder = train_FH_KF[-k_index,]
          rf_FH <- ranger(FH_target~.,data = remainder, num.trees = bframe[b,1],
num.random.splits = kframe[k,1], sample.fraction = lframe[l,1], min.node.size = nframe[n,1],
classification = T )
          prediction = predict(rf_FH,holdout[, -c(which(colnames(holdout) == 'FH_target'))])
          prediction = data.frame(prediction$predictions)
          colnames(prediction) = 'predictions'
          prediction$predictions = as.numeric(as.character(prediction$predictions))
          prediction$predictions = as.factor(prediction$predictions)
          holdout$FH_target = as.factor(holdout$FH_target)
          rfconf_mat_FH = caret::confusionMatrix(prediction$predictions, holdout$FH_target)
          rfconf_mat_FH = as.data.frame(t(rfconf_mat_FH$byClass))
          rf_FH_performance = rbind(rf_FH_performance,rfconf_mat_FH)
        }
        avg_rf_FH_performance = data.frame(t(colMeans(rf_FH_performance)))
        avg_rf_FH_performance$ID = iter
        avg_rf_FH_performance_overall = rbind(avg_rf_FH_performance_overall,
avg_rf_FH_performance)
        iter = iter+1
      }
    }
  }
}

plot_fh_rf = ggplot(data = avg_rf_FH_performance_overall, aes(x = ID, y = F1)) +
  geom_line()
plot_fh_rf

```

```

#RF TD
iter = 1
max_iter = nrow(bframe)*nrow(kframe)*nrow(lframe)*nrow(nframe)

train_TD = train[,-c(47,49:51)]
test_TD = test[,-c(47,49:51)]

#nn
rf_TD_performance = data.frame(t(conf_mat_TD$byClass))
avg_rf_TD_performance_overall = rf_TD_performance
avg_rf_TD_performance_overall$ID = 0
avg_rf_TD_performance_overall = avg_rf_TD_performance_overall[0,]
rf_TD_performance = rf_TD_performance[0,]
avg_rf_TD_performance = rf_TD_performance

# k_folds_set_index = sample(1:nrow(train_TD), 0.5*nrow(train_TD))

train_TD_KF = train[k_folds_set_index,-c(47,49:51)]
train_TD_KF[,c(1:24)] = lapply(train_TD_KF[,1:24], as.character)
train_TD_KF[,1:ncol(train_TD_KF)] = lapply(train_TD_KF[,1:ncol(train_TD_KF)], as.numeric)

for(b in 1:nrow(bframe)){
  for(k in 1:nrow(kframe)){
    for(l in 1:nrow(lframe)){
      for(n in 1:nrow(nframe)){
        print('% finished:')
        print(iter/max_iter*100)
        rf_TD_performance = rf_TD_performance[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_TD_KF),round(nrow(train_TD_KF)*1/3), replace = F)
          holdout = train_TD_KF[k_index,]
          remainder = train_TD_KF[-k_index,]
          rf_TD <- ranger(TD_target~.,data = remainder, num.trees = bframe[b,1],
num.random.splits = kframe[k,1], sample.fraction = lframe[l,1], min.node.size = nframe[n,1],
classification = T )
          prediction = predict(rf_TD,holdout[,-c(which(colnames(holdout) == 'TD_target'))])
          prediction = data.frame(prediction$predictions)
          colnames(prediction) = 'predictions'
          prediction$predictions = as.numeric(as.character(prediction$predictions))
          prediction$predictions = as.factor(prediction$predictions)
          holdout$TD_target = as.factor(holdout$TD_target)
          rfconf_mat_TD = caret::confusionMatrix(prediction$predictions, holdout$TD_target)
          rfconf_mat_TD = as.data.frame(t(rfconf_mat_TD$byClass))
          rf_TD_performance = rbind(rf_TD_performance,rfconf_mat_TD)
        }
      }
    }
  }
}

```



```

    avg_rf_TD_performance = data.frame(t(colMeans(rf_TD_performance)))
    avg_rf_TD_performance$ID = iter
    avg_rf_TD_performance_overall = rbind(avg_rf_TD_performance_overall,
    avg_rf_TD_performance)
    iter = iter+1
  }
}
}
}

#RF FS
iter = 1
max_iter = nrow(bframe)*nrow(kframe)*nrow(lframe)*nrow(nframe)

train_FS = train[,-c(47:50)]
test_FS = test[,-c(47:50)]

#nn
rf_FS_performance = data.frame(t(conf_mat_FS$byClass))
avg_rf_FS_performance_overall = rf_FS_performance
avg_rf_FS_performance_overall$ID = 0
avg_rf_FS_performance_overall = avg_rf_FS_performance_overall[0,]
rf_FS_performance = rf_FS_performance[0,]
avg_rf_FS_performance = rf_FS_performance

# k_folds_set_index = sample(1:nrow(train_FS), 0.5*nrow(train_FS))

train_FS_KF = train[k_folds_set_index,-c(47:50)]
train_FS_KF[,c(1:24)] = lapply(train_FS_KF[,1:24], as.character)
train_FS_KF[,1:ncol(train_FS_KF)] = lapply(train_FS_KF[,1:ncol(train_FS_KF)], as.numeric)

for(b in 1:nrow(bframe)){
  for(k in 1:nrow(kframe)){
    for(l in 1:nrow(lframe)){
      for(n in 1:nrow(nframe)){
        print('% finished:')
        print(iter/max_iter*100)
        rf_FS_performance = rf_FS_performance[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FS_KF),round(nrow(train_FS_KF)*1/3), replace = F)
          holdout = train_FS_KF[k_index,]
          remainder = train_FS_KF[-k_index,]
          rf_FS <- ranger(FS_target~.,data = remainder, num.trees = bframe[b,1],
num.random.splits = kframe[k,1], sample.fraction = lframe[l,1], min.node.size = nframe[n,1],
classification = T )
          prediction = predict(rf_FS,holdout[,-c(which(colnames(holdout) == 'FS_target'))])

```

```

prediction = data.frame(prediction$predictions)
colnames(prediction) = 'predictions'
prediction$predictions = as.numeric(as.character(prediction$predictions))
prediction$predictions = as.factor(prediction$predictions)
holdout$FS_target = as.factor(holdout$FS_target)
rfconf_mat_FS = caret::confusionMatrix(prediction$predictions, holdout$FS_target)
rfconf_mat_FS = as.data.frame(t(rfconf_mat_FS$byClass))
rf_FS_performance = rbind(rf_FS_performance, rfconf_mat_FS)
}
avg_rf_FS_performance = data.frame(t(colMeans(rf_FS_performance)))
avg_rf_FS_performance$ID = iter
avg_rf_FS_performance_overall = rbind(avg_rf_FS_performance_overall,
avg_rf_FS_performance)
iter = iter+1
}
}
}
}
}

```

```
#RF FTD
```

```
iter = 1
```

```
max_iter = nrow(bframe)*nrow(kframe)*nrow(lframe)*nrow(nframe)
```

```
train_HW = train[,-c(47,48,50:51)]
```

```
test_HW = test[,-c(47,48,50:51)]
```

```
#nn
```

```
rf_HW_performance = data.frame(t(conf_mat_HW$byClass))
```

```
avg_rf_HW_performance_overall = rf_HW_performance
```

```
avg_rf_HW_performance_overall$ID = 0
```

```
avg_rf_HW_performance_overall = avg_rf_HW_performance_overall[0,]
```

```
rf_HW_performance = rf_HW_performance[0,]
```

```
avg_rf_HW_performance = rf_HW_performance
```

```
# k_folds_set_index = sample(1:nrow(train_HW), 0.5*nrow(train_HW))
```

```
train_HW_KF = train[k_folds_set_index,-c(47,48,50:51)]
```

```
train_HW_KF[,c(1:24)] = lapply(train_HW_KF[,1:24], as.character)
```

```
train_HW_KF[,1:ncol(train_HW_KF)] = lapply(train_HW_KF[,1:ncol(train_HW_KF)], as.numeric)
```

```
for(b in 1:nrow(bframe)){
```

```
  for(k in 1:nrow(kframe)){
```

```
    for(l in 1:nrow(lframe)){
```

```
      for(n in 1:nrow(nframe)){
```

```
        print('% finished:')
```

```
        print(iter/max_iter*100)
```

```
        rf_HW_performance = rf_HW_performance[0,]
```

```

for(kf in 1:3){
  print(kf)
  k_index = sample(1:nrow(train_HW_KF),round(nrow(train_HW_KF)*1/3), replace = F)
  holdout = train_HW_KF[k_index,]
  remainder = train_HW_KF[-k_index,]
  rf_HW <- ranger(HW_target~.,data = remainder, num.trees = bframe[b,1],
num.random.splits = kframe[k,1], sample.fraction = lframe[l,1], min.node.size = nframe[n,1],
classification = T )
  prediction = predict(rf_HW,holdout[, -c(which(colnames(holdout) == 'HW_target'))])
  prediction = data.frame(prediction$predictions)
  colnames(prediction) = 'predictions'
  prediction$predictions = as.numeric(as.character(prediction$predictions))
  prediction$predictions = as.factor(prediction$predictions)
  holdout$HW_target = as.factor(holdout$HW_target)
  rfconf_mat_HW = caret::confusionMatrix(prediction$predictions, holdout$HW_target)
  rfconf_mat_HW = as.data.frame(t(rfconf_mat_HW$byClass))
  rf_HW_performance = rbind(rf_HW_performance, rfconf_mat_HW)
}
avg_rf_HW_performance = data.frame(t(colMeans(rf_HW_performance)))
avg_rf_HW_performance$ID = iter
avg_rf_HW_performance_overall = rbind(avg_rf_HW_performance_overall,
avg_rf_HW_performance)
iter = iter+1
}
}
}
}
}

```

```
#RF FT
```

```
iter = 1
```

```
max_iter = nrow(bframe)*nrow(kframe)*nrow(lframe)*nrow(nframe)
```

```
train_FT = train[, -c(47,48,49,51)]
```

```
test_FT = test[, -c(47,48,49,51)]
```

```
#nn
```

```
rf_FT_performance = data.frame(t(conf_mat_FT$byClass))
```

```
avg_rf_FT_performance_overall = rf_FT_performance
```

```
avg_rf_FT_performance_overall$ID = 0
```

```
avg_rf_FT_performance_overall = avg_rf_FT_performance_overall[0,]
```

```
rf_FT_performance = rf_FT_performance[0,]
```

```
avg_rf_FT_performance = rf_FT_performance
```

```
# k_folds_set_index = sample(1:nrow(train_FT), 0.5*nrow(train_FT))
```

```
train_FT_KF = train[k_folds_set_index, -c(47,48,49,51)]
```

```
train_FT_KF[, c(1:24)] = lapply(train_FT_KF[, 1:24], as.character)
```

```
train_FT_KF[,1:ncol(train_FT_KF)] = lapply(train_FT_KF[,1:ncol(train_FT_KF)], as.numeric)
```

```
for(b in 1:nrow(bframe)){
  for(k in 1:nrow(kframe)){
    for(l in 1:nrow(lframe)){
      for(n in 1:nrow(nframe)){
        print('% finished:')
        print(iter/max_iter*100)
        rf_FT_performance = rf_FT_performance[0,]
        for(kf in 1:3){
          print(kf)
          k_index = sample(1:nrow(train_FT_KF),round(nrow(train_FT_KF)*1/3), replace = F)
          holdout = train_FT_KF[k_index,]
          remainder = train_FT_KF[-k_index,]
          rf_FT <- ranger(FT_target~.,data = remainder, num.trees = bframe[b,1],
num.random.splits = kframe[k,1], sample.fraction = lframe[l,1], min.node.size = nframe[n,1],
classification = T )
          prediction = predict(rf_FT,holdout[, -c(which(colnames(holdout) == 'FT_target'))])
          prediction = data.frame(prediction$predictions)
          colnames(prediction) = 'predictions'
          prediction$predictions = as.numeric(as.character(prediction$predictions))
          prediction$predictions = as.factor(prediction$predictions)
          holdout$FT_target = as.factor(holdout$FT_target)
          rfconf_mat_FT = caret::confusionMatrix(prediction$predictions, holdout$FT_target)
          rfconf_mat_FT = as.data.frame(t(rfconf_mat_FT$byClass))
          rf_FT_performance = rbind(rf_FT_performance,rfconf_mat_FT)
        }
        avg_rf_FT_performance = data.frame(t(colMeans(rf_FT_performance)))
        avg_rf_FT_performance$ID = iter
        avg_rf_FT_performance_overall = rbind(avg_rf_FT_performance_overall,
avg_rf_FT_performance)
        iter = iter+1
      }
    }
  }
}
```

```
#RF FIN
```

```
#FH
```

```
bframe = data.frame('b' = c(500,550,600,650,700))
```

```
kframe = data.frame('k' = c(round(1/6*ncol(train)),round(1/4*ncol(train)),round(1/3*ncol(train))))
```

```
lframe = data.frame('l' = c(1/2,2/3,3/4,4/5))
```

```
nframe = data.frame('n' = c(3,5,7,9))
```

```
train_FH = train[, -c(48:51)]
```

```
train_FH[,c(1:24)] = lapply(train_FH[,1:24], as.character)
```

```
train_FH[,1:ncol(train_FH)] = lapply(train_FH[,1:ncol(train_FH)], as.numeric)
```

```

rf_FH_fin <- ranger(FH_target~.,data = train_FH, num.trees = 550, num.random.splits =
round(1/6*ncol(train)), sample.fraction = 3/4, min.node.size = 9, classification = T )
gc()
#TD
train_TD = train[,-c(47,49:51)]
train_TD[,c(1:24)] = lapply(train_TD[,1:24], as.character)
train_TD[,1:ncol(train_TD)] = lapply(train_TD[,1:ncol(train_TD)], as.numeric)
rf_TD_fin <- ranger(TD_target~.,data = train_TD, num.trees = 550, num.random.splits =
round(1/4*ncol(train)), sample.fraction = 2/3, min.node.size = 9, classification = T )
gc()
#FS
train_FS = train[,-c(47:50)]
train_FS[,c(1:24)] = lapply(train_FS[,1:24], as.character)
train_FS[,1:ncol(train_FS)] = lapply(train_FS[,1:ncol(train_FS)], as.numeric)
rf_FS_fin <- ranger(FS_target~.,data = train_FS, num.trees = 500, num.random.splits =
round(1/6*ncol(train)), sample.fraction = 3/4, min.node.size = 5, classification = T )
gc()
#HW
train_HW = train[,-c(47,48,50:51)]
train_HW[,c(1:24)] = lapply(train_HW[,1:24], as.character)
train_HW[,1:ncol(train_HW)] = lapply(train_HW[,1:ncol(train_HW)], as.numeric)
rf_HW_fin <- ranger(HW_target~.,data = train_HW, num.trees = 650, num.random.splits =
round(1/3*ncol(train)), sample.fraction = 3/4, min.node.size = 5, classification = T )
gc()
#FT
train_FT = train[,-c(47,48,49,51)]
train_FT[,c(1:24)] = lapply(train_FT[,1:24], as.character)
train_FT[,1:ncol(train_FT)] = lapply(train_FT[,1:ncol(train_FT)], as.numeric)
rf_FT_fin <- ranger(FT_target~.,data = train_FT, num.trees = 550, num.random.splits =
round(1/3*ncol(train)), sample.fraction = 3/4, min.node.size = 7, classification = T )

```

Appendix B

Variable conversion from long to wide Format

Table 47: Example of long format variable

| Bogie ID | Axle Number |
|-----------------|--------------------|
| 345 | 1 |
| 345 | 2 |
| 345 | 3 |
| 345 | 4 |

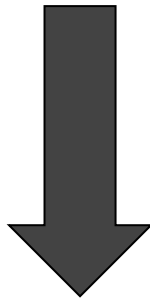


Table 48: Example of wide format variable

| Bogie ID | Axle_1 | Axle_2 | Axle_3 | Axle_4 |
|-----------------|---------------|---------------|---------------|---------------|
| 345 | 1 | 0 | 0 | 0 |
| 345 | 0 | 1 | 0 | 0 |
| 345 | 0 | 0 | 1 | 0 |
| 345 | 0 | 0 | 0 | 1 |